Benchmarks and Competitions in Termination, SAT and SMT

Workshop Symbolic Data, Leipzig, 2012

Johannes Waldmann [1]

December 14, 2012

[1] Fakultät IMN, HTWK Leipzig, Germany

Rewrite Systems and Termination

- rewrite system R: set of rules

```
(VAR x y) (RULES D(t) -> 1  D(1) -> 0
 D(+(x,y)) -> +(D(x),D(y))
 D(*(x,y)) -> +(*(y,D(x)),*(x,D(y)))
```

- rewrite step $\to_R$: apply rule to subterm
- rewriting is a (non-deterministic) model of computation
- application: programs on symbolic data (e.g., polynomials, programs, specifications ...)
- R is *terminating* $:= \to_R$ is well-founded (there is no infinite computation)

Termination Data Base & Competition

- goal: evaluation and comparison of tools that analyze termination of rewriting input: R, output: YES/NO + "proof" trace
- started in 2003 (as a session of the Workshop on Termination, 1993 ...)
- TPDB (termination problems data base): use common syntax for rewrite relations rules, strategy (innermost,...), theory (AC,...)
- Termcomp (termination competition): for each category, run all tools on all problems. executed on one dedicated server (then in Paris, now in Innsbruck), present results on web page

Termination Community & Infrastruct.

- Termination Workshop, Conference RTA
- by the way, *submit papers for RTA'13*, (Eindhoven, June 22–24; Deadline: February 1), we absolutely welcome reports on applications of rewriting (e.g., in Computer algebra)
- mailing list (mainly) for tool authors
- Wiki `http://termination-portal.org/` contains specifications, reports, links to data

current developments:

- better results data mining and presentation
- move to Star-Exec execution service

Competitions All Around

many computational logic communities with well-established competitions, e.g.,

- theorem proving (CASC, TPTP) `http://www.cs.miami.edu/~tptp/CASC/`
- boolean satisfiability (SAT)
- satisfiability modulo theories (SMT)

they all have domain-specific ...

- input syntax and semantics specification
- standards for what is an acceptable proof trace (from *none* to *informal* to *verifiable*)
- methods of selecting competition problems
- algorithms for scoring of results

Example: SAT

- input: a formula in propositional logic in conjunctive normal form
- output: a satisfying assignment
- typical method for solving:
  - DPLL (backtracking, with constraint propagation)
  - ... with CDCL (conflict driven clause learning)
- performs surprisingly well (despite NP-completeness)
- strong industry backing (motivated by circuit design verification, before production)
- benchmarks can be huge (10^6 variables)
- bi-annually: SAT competition, SAT race `http://www.satcompetition.org/`

Example: SMT

- given in advance: an algebra A
- input: quantifier-free formula F in predicate logic
- output: an assignment $\sigma : \mathrm{Var}(F) \to \mathrm{dom}(A)$ such that $\mathrm{value}(A, F, \sigma) = \mathrm{True}$
- examples
  - QF_LRA linear real arithmetic (boolean combination of linear inequalities over the reals)
  - QF_BV bitvectors (= machine numbers)
- strong industry interest (software verification), e.g., `http://z3.codeplex.com/`
- SMT-LIB, SMT-COMP (annual) `http://smtcomp.sourceforge.net/`
- application of SMT in termination analysis

The Star-Exec Project

`http://www.starexec.org/starexec/public/about.jsp`

- goal: provide a domain-agnostic exeution platform (software and hardware) for running competitions in computational logic
- directed by Starexec host (U.Iowa) and advisory board (representing the different communities)
- uses some meta-model (ontolgy) of competitions (benchmarks, tools, results)
- funding is secured, hardware is (partially) there, software is in beta state
- some open design issues, mainly on how much service to provide on the platform, and what to leave for the communities

Summary

- there is a wealth of experience with collecting and maintaining benchmarks
- mostly motivated by their use in competitions
- most of this is domain-specific
- Star-Exec tries to unify this over several domains (but so far it's not proven in practice)
- even if done for each domain separately: clear specification of semantics and syntax is tremendously useful
- (I think) benchmarks should be human-readable and -writable. (XML is bad, LISP is good)