rough diamond: how to get more out of your oracles

luís cruz-filipe
(joint work with kim s. larsen and peter schneider-kamp)

department of mathematics and computer science
university of southern denmark

itp 2017, brasília, brazil
september 28th, 2017

how it all started

sorting networks (itp'15)

very large computer proof that needed to be verified

- *ad-hoc* prolog program
- three weeks running time on 288 threads
- 28 GB trace allowing the proof to be replayed

how it all started

sorting networks (itp'15)

very large computer proof that needed to be verified

- *ad-hoc* prolog program
- three weeks running time on 288 threads
- 28 GB trace allowing the proof to be replayed

coq formalization

formalize a program to check the proof, use trace as "oracle"

- simplifies the formalization
- improves performance

making the checker usable

first results

able to check similar, smaller proofs

optimizations (cicm'15)

capitalize the fact that we are *rerunning* a proof

- analyse and optimize the data from the oracle
- change data structures, profit from meta-level properties

$\rightsquigarrow$ it is essential that we know the whole trace in advance!

and then we start to wonder...

are we on to something?

testing our hypothesis

find another large proof

characteristics:

- proof is known, needs to be checked
- contains several *existential subproblems*
- verification algorithm can be optimized from knowing the whole proof

a candidate: propositional unsatisfiability

the problem

verify a sat-solver's claim that a given propositional formula is unsatisfiable

(active topic of research, certified means not able to reach state-of-the-art)

a candidate: propositional unsatisfiability

the problem

verify a sat-solver's claim that a given propositional formula is unsatisfiable

(active topic of research, certified means not able to reach state-of-the-art)

our contribution (tacas'17)

follow the same approach

- formalize the relevant results in propositional logic
- directly implement a checking algorithm following the structure of the trace
- optimize the algorithm using knowledge about the whole proof

result of the experiment

- working prototype within two days
- two additional days of optimizations
- able to check all* results from the 2015 and 2016 sat competitions

(* all those covered by our format – the vast majority)

result of the experiment

- working prototype within two days
- two additional days of optimizations
- able to check all* results from the 2015 and 2016 sat competitions

(* all those covered by our format – the vast majority)

more interesting than sorting networks

- results immediately replicated by two independent groups
- three independent extensions to more expressive format

current contribution

a tentative methodology

- targets formalized proofs of results obtained by *ad-hoc* software
- identifies key characteristics required from the particular proof
- describes a working strategy

a rough diamond

- two case studies
- hard to evaluate formally
- potential for interesting applications

thank you!