Michael Whalen <mwwhalen@umn.edu>

ASE 2017 Paper Notification [351] - Reject

1 message

Massimiliano Di Penta and Tien N. Nguyen <ase2017-papers-chairs@borbala.com> Tue, Jul 18, 2017 at 1:04 AM
Reply-To: ase2017-papers-chairs@borbala.com
To: katis001@umn.edu, grigory@cs.washington.edu, andrew.gacek@rockwellcollins.com, guoxx663@umn.edu, whalen@cs.umn.edu, john.backes@rockwellcollins.com, agurfinkel@uwaterloo.ca
Cc: ase2017-papers-chairs@borbala.com, ase2017-papers-webadmin@borbala.com

Dear Andreas, Grigory, Andrew, Huajun, Michael, John and Arie,

Thank you for your submission to the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2017). We regret to inform you that your paper

"Validity-Guided Synthesis of Reactive Systems from Assume-Guarantee Contracts"

was not accepted for inclusion in the program of ASE 2017.

This year, the ASE conference received a record number of 388 submissions. Of these, 367 entered the reviewing process (314 technical papers, 8 experience papers, and 45 new ideas papers). Each paper has been reviewed and discussed by three or more program committee / expert review panel members, two of whom (at least) being program committee members. At the ASE Program Committee meeting in Santa Barbara on July 8-9, 2017, the program committee has made the following acceptance decisions:

- 88 contributions accepted, out of which 25 are conditional accepted (acceptance rate up to 24%);
- 65 technical papers accepted, out of which 13 are conditionally accepted (acceptance rate up to 21%);
- 2 experience papers accepted (25% acceptance rate);
- 10 new idea papers, one of which conditionally accepted (acceptance rate up to 22%);
- 11 technical papers conditionally accepted as new idea papers.

We appreciate your submission, encourage you to pursue further your research in this field, and eventually submit your results to future ASE conferences. We hope the reviews will be helpful to you.

We very much hope that you will still attend ASE 2017, which will be a very exciting event. ASE 2017 will take place in Urbana Champaign, Illinois, USA from October 30 to November 3, 2017. For details on the conference and the venue, please see the conference website at:

http://www.ase2017.org

Also, please note that submissions to ASE 2017 workshops are still open. Please have a look at http://ase2017.org/workshops

Sincerely,

Massimiliano Di Penta and Tien N. Nguyen
ASE 2017 Program Committee Co-Chairs

=--==--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=

>>> Summary of discussion leader <<<

This paper has been largely discussed and we agreed that the paper has some value, but the novelty from the theoretical point of view is not so convincing. Many people have studied and
implemented approaches to deductive synthesis. Some of these approaches have been successfully applied at scale to do interesting real world things.
The problems come when one tries to actually account for real world difficulties. People want a synthesis tool to produce correct-by-construction code that runs in

real world contexts where
the actual underlying machine models have all kinds of difficult limitations
(like finite bit-length numbers, floating point, finite memory sizes, arrays,
pointers, ...).
Unfortunately, the paper is not addressing these aspects.

=--==--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=

First reviewer's review:

>>> Summary of the submission <<<

The paper describes an algorithm for synthesizing reactive system
implementations that
conform to assume-guarantee specifications. Basically, it uses deductive
synthesis based
on the AE-VAL tool that does the heavy lifting, i.e., proving validity and
providing the
Skolem function that witnesses satisfiability which is straight-forwardly
translated
into the implementation. In the (typical) case when the assume part of the
contract is
not universally true, AE-VAL also returns a "zone of validity" formula, which
conditionalizes
the implementation. To reach a universal implementation, the authors'
algorithm
iterates over the results, each time cutting out the negation of this zone,
with iteration
continuing until a fixed point is reached.

>>> Evaluation <<<

I am unsure of the research novelty in this paper. It does cite Manna and
Waldinger's work
on deductive synthesis, but It seems to leave out Smith's work which used the
RAINBOW
prover to synthesize recursive programs. Later work form Kestrel (KIDS etc) has
also I believe
investigated reactive system synthesis and many practical real world synthesis
applications,
using a variety of techniques.

The idea of having a theorem prover prove a spec and produce a witness Skolem
to use in code
synthesis is an idea going back many, many years. The particular application of
this idea here
seems to be an interesting application of the idea, but I just don't see that
there is a new
"big idea" here.

Other comments:

* The example in Figure 1 seems to have a bug. The assumptions part needs to
include
 some constraints on the initial values of the b_i or else there is no
implementation.
 (e.g. if all the b_i start at 10, we lose immediately). The code implementing
this example
 later in the paper incorporates the assumption that the b_i start at 0, but
this should be
 in the figure as well in some form.

* How useful or significant is Theorem 2 in practice? It appears to be a
termination proof,
 but if we are using integers and linear reals in our theories and specs, then
merely
 cutting out "at least one state" at each iteration doesn't necessarily lead to
termination.
 So each iteration cutting out one state (Lemma 3) isn't too useful if it just
cuts out,
 e.g., "x=1", then "x=2" then "x=3", etc., ad infinitum.

Even in the case of finite data types (e.g. 32-bit integers), it could take extremely
long to converge, depending on details of the specs and theories.
As the authors have shown, the number of iterations taken by the
algorithm is of serious concern, and some of the Cinderella instances
failed to converge after multiple hours.

* Here are several issues that seem to me to be equally limitations of this work
as they are to other deductive synthesis applications.
(a) it is very hard (maybe too hard?) to write down a correct and complete spec
for a real world system.
(b) theorem provers (and related tools/functions) have difficulty automatically or even
semi-automatically producing the proofs needed for real world specs, especially
once the theories include integers and other rich data types.
(c) coming up with and being able to prove specs that soundly apply to real world
implementations having finite representations is far more complex than "simple"
infinite-integer reasoning. The authors do list this as an issue for future work,
but it seems to me this is central to any high-assurance safety-crticial application of
the work in the context of, say, an embedded system (avionics, medical equipment, etc).

It could be interesting to see a full industrial experience study of applying these techniques
in a particular domain, to see how the method in that domain can cope with the
above issues.

=--==--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*

Second reviewer's review:

>>> Summary of the submission <<<

This paper proposes a novel approach to automatically derive efficient implementations that are guaranteed to comply with given specifications. In the proposed approach safety specifications are wrote in the form of assume-guarantee contracts. The approach is inspired by inductive algorithms and attempts to reach a greatest fixpoint that contains states usable indefinitely by the system to reach unpredictable environments while guaranteeing the satisfaction of the specification.
The approach is tool supported and the tool is released as a branch of the Jkind model checker.
An experimentation with 110 contracts shows the effectiveness of the approach and improvements in performance with respect to an existing approach.

>>> Evaluation <<<

This paper presents a solid work that contributes to the program synthesis field. The overall formalisation is elegant and well written.
The experimentation shows that the approach is a promising contribution towards the synthesis of reactive system implementations from specifications expressed in linear integer and real arithmetic.

The audience of ASE would probably appreciate to have more details and discussion about the use of the approach in practice. This involves several aspects. First of all, authors might provide more information about the tool, how to use it, how to express the specification and what to do in order to have a program synthesised by using the approach.
Second, what kind of program is produced? Is the synthesised program ready to be used? What can be a practical context in which the approach can be used? Also, is the approach ready to be used in practical contexts? The authors say in the conclusion that the approach is a contribution “towards”, and this suggests that the approach is not ready yet. The question is, how far we are and what it is needed to be ready to be applicable in practice?

The fact of using assume-guarantee might lead to synthesise independently piece of software that could be easily composed. This might also support to re-synthesise only parts of the software when the specification evolves. It would be nice to see a discussion in this sense.

Strengths and weaknesses

+ nice and solid work
+ well-written paper
- the paper might be improved through a description of its usability in practice

=--==--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*

Third reviewer's review:

>>> Summary of the submission <<<

The paper presents a novel approach to the synthesis of reactive programs starting from safety specifications in the form of assume/guarantee contracts. The approach is based on the idea of incrementally pruning out of the problem space those regions that violate the contract until a fixpoint from which a solution fulfilling the contract can be derived. The evaluation shows that the proposed approach improves the performance of a similar one and it is also able to find a solution for some cases that are not solvable by the other. For these cases authors provide also a comparison with another approach that finds a solution starting from templates. compared to this, the proposed approach provides lower performance.

>>> Evaluation <<<

The paper is relevant to ASE, but the details of the proposed approach could be better appreciated in a more specialistic venue. It is also well organised and can be followed (with some difficulties) even by a non expert reader.
The proposed approach is adequately presented. However, the development of an approach to deductive synthesis is not new per se. The open challenge in the area is the development of a synthesis tool that is able to fully address aspects such as pointers, arrays and the like.
However, these aspects do not seem to be completely addressed in the paper.

The initial description of Cinderella's contract is not clear. In particular, authors do not explain what "e" is. This will become clear later on in the paper, but even with that explanation, I do not understand why it appears to be described as one of the outputs (outgoing arrow in figure 1) of Cinderella implementation. Also, in the text the term "ite((e = k..." appears without an explanation of what "ite" means.
In Section IV.D authors write that the validity-guided algorithm enters in an infinite loop. This is not correct, at least for finite models (lemma 3) and is misleading in that part of the text that is describing the application of the algorithm to a case that terminates.
The evaluation considers a number of benchmarks that, however, are not described in the paper, even though they are available on a github repository. As such, it is difficult to appreciate the effectiveness of the approach on these cases.

Summary of strengths and weaknesses
+ reasonably well written paper
+ useful approach
- the novelty of the approach is not clear
- benchmarks used in the evaluation are not described

=--==--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*=--=*