

From: ESOP 2017 esop2017@easychair.org
Subject: ESOP 2017 review response (submission [“NUMBER”])
Date: 7 December 2016 at 11:40
To: Susan Eisenbach s.eisenbach@imperial.ac.uk



Dear Susan,

Thank you for your submission to ESOP 2017. The ESOP 2017 review response period will be between 7 Dec 2016 and 9 Dec 2016 (anywhere on earth).

During this time, you will have access to the current state of your reviews and have the opportunity to submit a response. Please try to be succinct and form your response within 500 words. Reviewers are likely to stop reading your response after this 500-words limit. The hard limit is 1000 words.

Please keep in mind the following during this process:

- * The response must focus on any factual errors in the reviews and any questions posed by the reviewers. It must not provide new research results or reformulate the presentation. Try to be as concise and to the point as possible.
- * The review response period is an opportunity to react to the reviews, but not a requirement to do so. Thus, if you feel the reviews are accurate and the reviewers have not asked any questions, then you do not have to respond.
- * The reviews are as submitted by the PC members, without any coordination between them. Thus, there may be inconsistencies. Furthermore, these are not the final versions of the reviews. The reviews can later be updated to take into account the discussions at the program committee meeting, and we may find it necessary to solicit other outside reviews after the review response period.
- * The program committee will read your responses carefully and take this information into account during the discussions. On the other hand, the program committee will not directly respond to your responses, either before the program committee meeting or in the final versions of the reviews.
- * Your response will be seen by all PC members who have access to the discussion of your paper, so please try to be polite and constructive.
- * Some reviews of your paper might be missing. I apologise for this delay. I will do my best to send these reviews as soon as possible.

The reviews on your paper are attached to this letter. To submit your response you should log on the EasyChair Web page for ESOP 2017 and select your submission on the menu.

Best wishes,
Hongseok

----- REVIEW 1 -----

PAPER: 105

TITLE: Modular Verification of Procedure Equivalence in the Presence of Memory Allocation

AUTHORS: Tim Wood, Shuvendu K. Lahiri, Sophia Drossopoulou and Susan Eisenbach

OVERALL EVALUATION: 1

REVIEWER'S CONFIDENCE: 4

----- Review -----

The paper considers the problem of checking when two procedures have equivalent behaviour. This is done for a programming language with dynamic memory allocation (but without pointer arithmetic) and recursive procedures. The technical contributions are two-fold. First, the authors propose a theory showing that, when reasoning about the procedure correspondence, instead of reasoning about isomorphism between heaps of the two procedures, we can often assume them to be exactly equal for free. This is an intuitive property for a language where programs are insensitive to memory addresses. The second contribution is an implementation of this theory in a Boogie-based tool for checking procedure equivalence. The key idea is to establish a correspondence

between allocation sites so that the memory cells allocated at matching sites play the same role. Then the theory proposed by the authors allows assuming that the addresses allocated at the matching sites are the same. The SMT solver is then used to search for a good correspondence between allocation sites. To handle recursive procedures, the authors' theory allows assuming a correspondence between recursive procedure calls when establishing a correspondence between procedures.

Pros:

- The result about assuming heap equality is nice and intuitive.
- Tool support for a theory is a plus. The approach of matching allocation sites is simple, but pragmatic.


Cons:


- Even though the authors describe the implementation of the theory in a tool, they provide absolutely no evaluation of the tool (and there's still space in the paper!). What kind of examples was the tool able to handle successfully? I find the absence of evaluation very strange and also worrying, because the SMT-based tool requires using triggers, frame axioms, etc., and it's unclear what amount of manual effort is required to verify actual examples. The soundness result also assumes constraints on procedure termination, and it's unclear whether the tool checks them. It would be nice if the authors could comment on this in the rebuttal.
- The authors state the requirements that a tool using their theory has to satisfy (Definition 5), but never argue that their tool actually satisfies these requirements. I'm worried about this because item 2 in Def 5 requires establishing an isomorphism between heap regions. What guarantees the existence of this isomorphism? Is it the success of the tool? The authors give only a very vague explanation (before Def 5) of what their tool does. E.g., what do you mean by "interesting correspondences"? Also, what is π in $\text{in}(\pi \cup \text{Pi}(\text{tr1}, \text{tr2}))$ in item 3? In general, I found Def 5-6 poorly explained. Again, the authors should feel free to expand in the rebuttal.

Additional comments:

I found the presentation somewhat suboptimal. The Boogie programs on which the authors rely in explanations are messy and hard to follow, at least for someone not familiar with the syntax.

The description of mutual summaries in Section 2.3 was hard to follow.

Section 3.1: L never ... reuses allocated memory - that's an issue. A garbage-collected language will actually reuse memory. How difficult is it to lift this constraint? 

Section 3.7. It would be better if you provided some intuition for why mutual termination requirement is needed. Also, looks like is this something your tool has to ensure - does it? 

Section 4.2. It would be better to explain the reachability axioms instead of just dumping the Boogie code. Also, I guess the (incomplete) axiomatisation is tailored to prove certain kinds of examples, but the absence of evaluation makes it impossible to judge how effective it is. You say that axiom instantiations are "controlled by various triggers", so is user interaction required in this case?

----- REVIEW 2 -----

PAPER: 105

TITLE: Modular Verification of Procedure Equivalence in the Presence of Memory Allocation

AUTHORS: Tim Wood, Shuvendu K. Lahiri, Sophia Drossopoulou and Susan Eisenbach

OVERALL EVALUATION: 2

REVIEWER'S CONFIDENCE: 4

----- Review -----

This paper presents a method for automatically verifying the equivalence of procedures that allocate memory and make use of garbage collection.

The contribution of this paper over previous equivalence checkers is the ability to consider procedures equivalent up-to isomorphic memory states. Up-to isomorphism is required in order to deal with

inessential differences in the order that fresh memory locations are allocated. The method presented attempts to reduce isomorphism checking to equality checking by trying several different ways of equating the results of the allocation instructions between the two procedures. A soundness proof for the technique is given, and the encoding of the approach in an SMT solver (via Boogie) is discussed.

I really liked this paper. The setting and practical applications are clear. The problem is clearly explained with a revealing example. The solution is well presented, and I think the main idea is intuitively appealing. The technical body of the paper (Section 3) is relatively straightforward to follow, modulo some omissions in minor definitions.

My only major problem with this paper is that there is not much in the way of experimental evaluation of the approach. There are only two examples given, one which succeeds, and one which exposes an incompleteness of the approach. I would've expected to see mention of more examples of the tool in action. And possibly some rough timing measurements.

Nevertheless, I think this paper is well-written and provides a good contribution over existing procedure equivalence checking techniques. Therefore I think it ought to be accepted.

In summary, the plus and minuses of this paper are:

- + Seems like a real advance over previous equivalence checkers that couldn't handle memory states up-to iso.
- + The approach is proved to be sound, and given an axiomatisation that works effectively with an SMT solver (Z3)
- + The approach of attempting to reduce isomorphism checking to equality checking by guessing some equalities looks reasonably simple, and could possibly be used in other systems.
- While there is quite a bit of discussion about related approaches at a theoretical level, including relational program logics and so forth, there isn't any evaluation of the tool itself. Given that quite a bit of the paper is spent discussing effective ways of encoding the necessary axioms in an SMT solver, it is a bit disappointing that there is only one example of a pair of procedures that APE/RIE can prove equivalent, and one example of the tool's incompleteness.
- It wasn't clear to me, to what extent are contracts for the procedures in question required? Are 'modifies' sets required?
- There are quite a few little typos and unexplained bits of notation (some listed below). Individually, these are not that much of a problem, but they do add up to make Section 3 (Soundness) a bit harder than it need be.

Other comments:

Page 3, 2nd last para: "of not" => "or not". Also, this paragraph repeats quite a bit of what the caption of Fig 1 says about the difference between 'lcopy' and 'rcopy'.

Page 7: "The solver is unable to directly prove that the reachability sets are the same on several examples." I'm not sure what this sentence is saying. Do you mean that for the "obvious" but unstated definition of equality of reachable sets, there are examples where Z3 cannot prove equality? Why not tell us the examples?

TW rewrote this a bit

Page 8: "the predicate \$Heap#EqualFromParams [...] detailed in Section 2.2". I don't see this mentioned in Sec 2.2?

now done

Page 8, lines 78 and 79 of the figure: are there missing \$strat arguments?

done

Page 9, 4th para: "detailed" => "detail".

I disagree

Page 10/11: The text does not explicitly explain the relationship between the curved arrow and the zig-zaggy arrow with a subscript. It

took me a while to realise that one was for atomic statements, and one for compound statements.

Page 11, bottom right rule in the semantics: '&&' should be '\land'

done

Page 12: The text refers to a rule "BODV", but the figure has a rule called "BOD". Does "BODV" specifically refer to the 'V' variant?

Page 12: "as the chapter progresses" Is this a cut-n-paste error from a PhD thesis chapter? Could just say 'Sections 3.5 and 3.6'.

Page 13: "We use the notation that:" is a bit awkward.

Page 14: I don't see where σ^{top} is defined (refers to the topmost stack frame?). Also, in the paragraph before Defn 3, I think "sequences of parameter *names* W and X" would be clearer.

now done

Page 15: I think that inserting " $\text{tr1} \approx \text{tr2}$ " after the first "isomorphic" would be clearer. Otherwise, it is not clear where the π in the first sub-point comes from.

Page 23: "unbalance" \Rightarrow "unbalanced"

----- REVIEW 3 -----

PAPER: 105

TITLE: Modular Verification of Procedure Equivalence in the Presence of Memory Allocation

AUTHORS: Tim Wood, Shuvendu K. Lahiri, Sophia Drossopoulou and Susan Eisenbach

OVERALL EVALUATION: 2

REVIEWER'S CONFIDENCE: 3

----- Review -----

This paper proposes a methodology called RIE for automatically proving equivalence of programs performing dynamic memory allocation on heap data structures. The goal is to capture equivalence of allocations performed in different orders and modulo garbage. Given two procedures to be proved equivalent, the technique is to consider all possible permutations of allocation points within these two procedures. The tool designed by the authors, APE, is producing a Boogie programs that encode all these permutations, so that the SMT solver Z3 will figure out if out of these establishes the equivalence of heaps at the end of the two procedures.

Even if there are obvious limitations to the method, this is a nice paper. It reads easily; it addresses an important and challenging problem; it proposes a new, sound methodology; and it comes with a prototype implementation.

Though the paper has room for improvements (see my comments below), it could make a nice contribution to ESOP and I'm in favor of acceptance.

Note: a footnote explains that this is a short version of the first author's (Tim Wood) PhD, which is referred to for more details. But reference [45] is not providing any link to the PhD (it simply says "Under submission"), which cannot be found either on the author's web page. Too bad.

Detailed comments:

- p2, "our definition of equivalence": I don't get how "they result in isomorphic stores" and "allows for differences in the order of amount of memory allocation" are not in contradiction? If we allocated more, how can we get an isomorphic store? Does this mean "we allocated more but we have freed the exact amount of extra memory before the procedure ends"?
- p3, challenge 3: isn't this obvious?
- p3, Fig 1: please explain the field v used in the code
- p3, Fig 1: there is a user annotation (modifies {r}), which contradicts the sentence "without programmer annotations" from page 1.
- p3: what is APE? (first time mentioned here)
- n4: "are allocated different addresses" ?

- p4: do not typeset RIE in italics ✓
- p5: mentioning "modulo garbage" (end of the page) helps. I wish this formulation had been used earlier in the paper.
- p6: "fig. 2" -> "Fig. 2" Generally speaking, references to figures, sections, theorems, etc. should be capitalized. Please check for other occurrences in the paper (there are some). ✓
- p7: footnote 6 is missing a final dot. ✓
- p7: it is a bit annoying that Fig. 3 is referring to $\$Reach$ that is only defined in Sec. 4.2. When reading Fig. 3, I was definitely curious about the way $\$Reach$ was defined and I had to jump to page 21 to read the whole Sec. 4.2 before returning to page 7. *added a comment*
- p8: ", this is overcome by" -> do not separate two sentences with a comma (there are other occurrences of such a mistake in the paper) ✓
- p9: the third paragraph ("The framing axioms...") should be merged with the one ending on top of page 9. ✓
- p9: "mutual-summaries" -> no dash ✓
- p9: triggers are mentioned ("We set triggers") before being explained (in the next paragraph)
- p9, footnote 7: "but is not checked" -> there is no code associated to these procedures, so what would have to be checked here?
- p10: It is inelegant (and typographically debatable) to typeset First, Second, ..., Seventh in bold font. ✓
- Fig 5: in the grammar rule for $ScalarExpr$, "b" should be typeset in italics. **done**
- Fig 5: if $BoolExpr$ is included in $ScalarExpr$, why having both " $x := e$ " and " $x := b$ " in $AtomicStmnt$? And by the way, there is a semantics rule ASSIGN for $x:=e$ but no rule for $x:=b$. **fixed first part**
- Fig 5: the rule for conjunction uses $\&\&$ instead of \wedge
- Fig 5: missing rules for Boolean expressions "true" and "false", and missing rules for deriving $signal=b$. I would rather expect first a set of complete rules to define $[[b]]\sigma$ for a Boolean expression, and then a definition of $signal=b$ as a shortcut for $[[b]]\sigma = true$.
- Fig 5/6: I don't get the definition and use of $mkframe$. It seems that names for formal and actual parameters are identified.
- Fig 6: what is body p in rule CALLV? It should be instead $\{cal B\}(p)$ I guess.
- Fig 6: what is σ_{1} in rule CALLA?
- p12: "We define I and M, as the chapter progresses." -> "section" instead of "chapter"? (surely a copy-paste from the thesis) More seriously, it is not ideal to have a rule in Fig 6 using two predicates (I and M) defined much later. ✓
- p13, footnote 10: this is not the definition of an injection. That should rather be the weaker property " $b=d \Rightarrow a=c$ ".
- p13, Def 1: null, true, and false should not be typeset in italics (and too many space in the typesetting of "false" suggests it is incorrectly typeset in math mode)
- p13, Def 2: the type of "Con" should be given earlier (that would help understanding Sec. 3.1).
- p14: typesetting $\{cal L\}$ on top or below other symbols (such as equivalence or semantics arrows) is rather ugly.
- p15: "don't" -> "do not"

p15: where is Π defined? (sorry if I missed that)

- p17: "there exist an"

- p18: "doesn't" -> "does not"

- p19: "naïve" -> naive (unless you want to spell it as in French, but then it is "naïve")

- p19, near bottom of the page: avoid symbols at the beginning of a line (here "rf"). Use LaTeX's unbreakable space ~ for that purpose.

- p20: caption for Fig. 7 is missing a final dot. (Please check other captions.)

- p21: I don't understand the last sentence of Sec. 4.1.

- p21: I don't get why the technical Sec. 4.2 comes so late in the paper and is included in an overall section called "Discussion".

- p21, Fig. 8: the caption mentions "\$Allocated" but the figure is not using it at all!

- p21, Fig. 8: there are four axioms, but actually one should be able (in principle) to deduce the second and fourth from the other two. So they should be lemmas instead. (The last sentence of this section suggests that the authors are aware of that.)

More importantly: this axiomatization of $\$Reach$ with four axioms is not capturing that $\$Reach$ is the *smallest relation* defined by this set of axioms. Please explain why this is not an issue here.

- p21/22: naturally, I would think that (dis)proving reachability properties requires induction. Since SMT solvers are not performing any proof by induction, this should be clarified in the paper.

- p22: "post conditions" -> postconditions

- p23, related work: there is a bit of duplication between this section and the introduction (page 2). I suggest removing some related work discussion from the introduction.

- p23: "built in support" -> built-in support

- p24: "data-structures" -> data structures
