

Towards Pratical Abstract Execution

ISoLA 2022

Detailed Report

Typos

- ~~General:~~

~~The title of the paper and the titles of the sections should be in title case, e.g. “Towards practical abstract execution” should be “Towards Practical Abstract Execution”~~

- Page 1

~~line 9: type E_{boot} instead of $E_{boot\text{ean}}$~~

~~line 19: reference to paper [29] (Abstract Execution 2019), instead of [27] (REFINITY 2020)~~

~~line 20: reference to paper [27] (REFINITY 2020), instead of [1] (The KeY Book 2016)~~

- Page 2

~~line 13: “exceptions and object” instead of “exceptions and objects”~~
~~line 14: the “resp.” does not respect the order between “exceptions” and “objects”~~

- Page 3

~~line 28: comma after closed bracket should be removed~~

~~line 32: “contents” should be “content”~~

list of things
that needs to be
added for APEs

- Page 5

~~Listing 2 and 3 have the same caption~~

~~line 1: type “abst[r]act”~~

we describe the ref. to/for REFINITY

~~line 4: “refactoring to REFINITY”, possible wrong preposition “to”~~

- Page 6

~~line 3: “any exceptions” instead of “any exception”~~

~~line 29: inbetween instead of in between (in between)~~

~~line 37: “an behavioral” instead of “a behavioral”~~

- Page 8

~~line 14: “REFINITY did” should be “REFINITY does”~~

- Page 9

~~line 18: “global implicit counter” should be “implicit global counter”~~

~~lines 29,30 (Definition 2): the σ in $\sigma' \prec_h^C \sigma$ and $\sigma \preceq_h^C \sigma''$ should be σ_h^C~~
don't like this change, but done

- Page 10
 - ~~– Listings 6 and 7 have the same caption~~
- Page 12
 - line 30: no comma after closed bracket
- Page 13
 - ~~– line 13: missing period after “equivalences”~~
- Page 14
 - ~~– line 19: “that both programs” should be “that the two programs”~~
 - ~~– line 20: not clear whether “the corresponding” should be “correspondence” or whether a noun is missing~~ went with a correspondence
- Page 15
 - line 15: “is that due to” should be “is due to”
 - line 23: “Adressing” instead of “Addressing”, “would requiring” instead of “would require”
 - line 32: “lacks placeholders ” should be “lacks of placeholders”
 - line 40: incomplete sentence starting with “due to”, the main sentence is missing
- Page 16
 - line 20: “similarly are” should be “similarly were”
 - line 30: add “and” after “symbolic execution”
 - line 31: replace semicolon with comma
 - line 39: “syntactical different” should be “syntactically different”
- Page 17
 - line 6: not clear whether “compose to specify” should be “allow to specify” or else
 - line 8: replace “and make use” with “by making use”
- Page 18
 - line 1: “Refnity” instead of “REFINITY”
 - line 8: replace “and make use” with “by making use”
- Page 20
 - Paper referred by [28] and [29] is the same

Presentation

- Page 1
 - line 9: The example shown presents a refactoring that would be behavior preserving only if the statement was followed by a void **return** statement. This should be clarified.
- Page 2
 - lines 5-8: it should be reformulated and should be clearer that it is a research question
 - lines 39-40: this two lines should be reformulated and come after the section about symbolic execution
- Page 3
 - line 9: “then branch”, “then” should be highlighted
 - line 11: “else branch”, “else” should be highlighted
 - lines 4-11: the sentence is too long.
 - lines 33-34: the sentence starting with “Only that [...]” should be reformulated, and it should not start with “Only that”
 - lines 35 and 36: “wish to specify [...]” “achieve such a specification” seem in contradiction, it would be better “want to specify”
 - line 38-40: JML should be cited here, the presentation should be more straightforward.
 - line 42: formulation should be changed to “abstract statement N may assign to [...], and access [...]”
- Page 4
 - line 1: replace “placed” with a more suitable verb.
- Page 8
 - line 2: refer to figure in “when compared to the sketched out example above”
 - line: 13: “[...] of return values **of the sides** being identical [...]” is unclear and should be reformulated, possible using “sides” only when referring to figures/listings, otherwise it is not clear what “sides” refers to.
 - line 24: “**as must be** the objects” should be rewritten “**as well as** the objects”
- Page 9

- Adding names for definition 1 and 2 would help the reader to understand them better
- line 14: the sentence starting with “Continuing our investigation [...]” should be rewritten to be less redundant and to make clear how the side effects in constructors is crucial in proving correct a refactoring involving object creation.
- Page 10
 - line 10: make clear that the sentence “This suffices to prove [...]” holds only if the side effects visible by the constructor of \mathbf{C} are not visible by the constructor of \mathbf{D} and vice versa.
- Page 11
 - line 25: “Again, we assume that the constructor of \mathbf{C} has no side effects except object creation on the heap”, why “Again”? This is the first time you present such a restrictive constrain. Until now the constrain for the constructors was about the visibility of side effects.
- Page 13
 - line 9-13: sentence starting with “The programs are [...]” should be rewritten to make clearer the conclusion.
- Page 17
 - line 8: remove sentence “tackle [...] make” and replace “use of an infrastructure” with “use an infrastructure”: the resulting sentence would be “They use an infrastructure to [...]”
 - lines 28-32: rewrite this sentence, it is too long and it has no punctuation.
 - lines 37-41: rewrite this sentence, it is too long and it has almost no punctuation.
 - line 37: put round brackets around “sometimes use-case specific”
 - line 40: “confounding” should be “confusing”

Content

- Page 2
 - line 17: reduces the size specification compared to what?
- Page 3
 - lines 16-18: wrong citation to [1] (3.3)

*“Semantically JavaDL formulas are **not** evaluated in a Kripke structure **over** a collection of first-order structures”*

while the original is:

“On the semantic level, the difference is that JavaDL formulas are not evaluated in a single first-order structure but in a so-called Kripke structure, which is a collection of first-order structures.”

- line 32: “[...] we do not want to specify the exact contents of its method body”, the content has always to be specified exactly, but using AE it is possible to define it (partially) abstractly.

- Page 6

- line 1: it would be interesting to see the code of the working example

- Page 8

- line 2: “which contains no surprises”: what does it mean?
- line 11: “published version of REFINITY”, which one?