Unrealizability Logic Corrigenda

Jinwoo Kim

This document serves as a corrigenda for the POPL 2023 paper "Unrealizability Logic", for which the authors discovered a mistake related to soundness and completeness in the paper after publication.

We will start with a list of affected claims made in the paper, then describe each item in detail.

A list of affected claims

In the paper, we claimed the following:

- Unrealizability logic is sound even when the grammar contains loops.
- Unrealizability logic is complete even when the grammar contains loops, through use of the `While` rule.

The authors discovered that the claim about soundness has an assumption that was not made clear in the paper. We clarify this assumption, and then give an update to the `While` rule that allows soundness to hold even without the assumption.

We then issue a corrigendum about completeness: the completeness proof in the original paper is flawed, and thus we update unrealizability logic with a non-compositional rule to show completeness.

We note that the corrigenda do not affect unrealizability logic in the case where the target language is loop-free: in this case, unrealizability logic remains sound and complete following the arguments in the paper.

Corrigenda

Soundness

In Section 3.3, page 16 of the paper, we give the following rule for while loops:

$$\frac{\begin{array}{l}\Gamma \vdash \{\!\!\{\mathcal{I}\}\!\!\}\ B\ \{\!\!\{\mathcal{I}_B\}\!\!\} \\ \Gamma \vdash \{\!\!\{\mathcal{I}_B \wedge \mathbf{b_{loop}} = \mathbf{b_t}\}\!\!\}\ S\ \{\!\!\{\mathcal{I}'_B\}\!\!\}\end{array} \qquad \begin{array}{l}\mathbf{b_{loop}}, \mathbf{v_1}, \mathbf{v_2} \text{ fresh} \\ \exists \mathbf{v_1}, \mathbf{e_t}, \mathbf{b_t}. \mathcal{I}'_B[\mathbf{v_1}[i]/\mathbf{v}[i] \text{ where } \mathbf{b_{loop}}[i] = \text{false}] \implies \\ \exists \mathbf{v_2}, \mathbf{e_t}, \mathbf{b_t}. \mathcal{I}_B[\mathbf{v_2}[i]/\mathbf{v}[i] \text{ where } \mathbf{b_{loop}}[i] = \text{false}]\end{array}}{\Gamma \vdash \{\!\!\{\mathcal{I}\}\!\!\} \text{ while } B \text{ do } S\ \{\!\!\{\mathcal{I}_B \wedge \mathbf{b_t} = \text{f}\}\!\!\}}\ \textsc{While}$$

In Section 4, Theorem 4.1, we claim that unrealizability logic is sound. This theorem about soundness has an additional assumption that the predicates in question are not *hyperproperties*, that is, properties that must be described via a predicate that relates multiple different examples. An example of such a hyperproperty is injectivity, which in unrealizability logic, can be described by the triple $\{\forall i, j.\, x_i \neq x_j\}\ S\ \{\forall i, j.\, x_i \neq x_j\}$.

We make the following clarifications:

- Unrealizability logic is sound if the pre- and post predicates do not describe hyperproperties.

- If the predicates do describe hyperproperties, then only the original `While` rule becomes unsound in the sense that it may derive an invalid triple as a conclusion, even when given only valid triples as premises. The rest of the rules remain sound (i.e., they will only derive valid conclusions from valid premises).
- The original paper only considers synthesis over non-hyperproperty specifications, as defined in Definition 3.2. We thus first clarify that there is an extra assumption to Theorem 4.1 that the predicates are not hyperproperties.
- We observe that because the rules of unrealizability logic *do not introduce* hyperproperties, one will not encounter hyperproperties as part of an unrealizability logic proof, assuming that hyperproperties are absent from the specification itself.
- However, it is also true that it is possible and desirable for unrealizability logic to support specifications that are hyperproperties as well. **We thus give an alternative formulation of the `While` rule that is sound for hyperproperties.**

$$\frac{\begin{array}{ll}\mathbf{b_{loop}}, \mathbf{v_1}, \mathbf{v_2} \text{ fresh} & (\exists \mathbf{v_1}, \mathbf{v_2}, \mathbf{v_1'}, \mathbf{v_2'}, \mathbf{b_t}. \mathcal{I}_B'[\mathbf{v_1'}[i]/\mathbf{v}[i] \text{ where } \mathbf{b_{loop}}[i] = \text{false}] \wedge \\ \Gamma \vdash \{\!|\mathcal{I}|\!\}\ B\ \{\!|\mathcal{I}_B|\!\} & (\mathcal{I}_B \wedge \mathbf{b_{loop}} = \mathbf{b_t} \wedge \mathbf{v} = \mathbf{v_2})[\mathbf{v_2'}[i]/\mathbf{v}[i] \text{ where } \mathbf{b_{loop}}[i] = \text{true}] \wedge \\ \Gamma \vdash \{\!|\mathcal{I}_B \wedge \mathbf{b_{loop}} = \mathbf{b_t} \wedge \mathbf{v} = \mathbf{v_1}|\!\}\ S\ \{\!|\mathcal{I}_B'|\!\} & (\mathbf{v_1} = \mathbf{v_2})) \implies (\exists \mathbf{b_t}. \mathcal{I}_B)\end{array}}{\Gamma \vdash \{\!|\mathcal{I}|\!\} \text{ while } B \text{ do } S\ \{\!|\mathcal{I}_B \wedge \mathbf{b_t} = \text{f}|\!\}}\ \text{While\_Hyp}$$

- `While_Hyp` is an updated version of the original `While` rule that preserves soundness on hyperproperties. The explanation of `While_Hyp` is as follows:
  - Like the rule for loops in Hoare Logic, `While_Hyp` depends on the existence of an invariant I that, in the case of unrealizabilty logic, works for all possible completions of the loop body S.
  - Because unrealizability logic operates over vector-states, the condition that checks whether I is an invariant (the right-hand premise of `While_Hyp`) is much more involved. First, we take a similar approach as in the `ITE` rule and extend the invariant I_B with the ghost state $\mathbf{v_1}$, then push it through the body S regardless of the loop guard.
  - Having obtained I'_B in this way, the invariant is checked by first creating the set of vector-states for which the entries on which the loop guard evaluates to True are preserved, and the entries on which the loop guard evaluates to False are recovered to the states prior to executing the loop body (achieved by the premise of the implication on the right). The two sets of entries are synchronized via checking equality between the ghost states $\mathbf{v_1} = \mathbf{v_2}$, similar to the `Plus` rule in the original paper. Then we check that this set is a subset of the original invariant I_B.

To see that `While_Hyp` can only derive valid triples even when the predicates describes hyperproperties, consider a vector-state $\sigma \in I$. For each $b \in B$ and $s \in S$, if σ terminates on $\mathsf{While}\ b\ \mathsf{do}\ s$, $\mathbf{b_t} = f$; also note that $I = I_B$ on all variables outside of b_t, and that the implication on the right of the rule denotes that the result of executing σ on vector-entries where the loop guard evaluates to `True` is a subset of I_B. Because I is an invariant on program variables, the result of executing any number of iterations of S on the appropriate entries of a vector-state σ is guaranteed to fall within I_B; combine with the argument with the loop guard to obtain the conclusion of the rule.

The original rule `While` checked the invariant condition only for examples where the loop guard evaluates to `True`. This approach is fine for a non-hyperproperty setting, because examples that do not pass the loop guard are guaranteed to satisfy the invariant, but fails for hyperproperties, because the relation between examples that pass and do not pass the loop guard changes.

Completeness

In Section 4, page 20, in the proof of Lemma 4.3, we claim that unrealizability logic is complete using the rule `While`, assuming that the weakest precondition of a set of loops $wp(\textsf{While } B \textsf{ do } S, Q)$ on an arbitrary postcondition Q is expressible in the assertion language of choice. This statement about completeness is what must be corrected: the expressibility of the weakest precondition is not enough to guarantee completeness (whether using the original `While` rule (for non-hyperproperties), or the updated version `While_Hyp` when considering hyperproperties). Instead, completeness through `While` or `While_Hyp` is preserved if, for any P and Q, there exists a predicate I satisfying the following conditions:

- $I \implies wp(\textsf{While } B \textsf{ do } S, Q)$ and I is an invariant of (the set of programs) S on those examples where the loop guard is satisfied.
- Denoting the result of executing I through (the set of loop guards) B as I_B, $I_B \wedge \mathbf{b_t} = \mathsf{f} \implies Q$.

However, such a predicate I does not exist in general when the vocabulary of predicates are limited to program and auxiliary variables as done in the original paper. On the other hand, if predicates are allowed to introduce additional variables that capture auxiliary information (such as the structure of the loop body), proving or disproving the existence of an adequate invariant becomes a nontrivial task.

We thus give another rule for loops in addition to `While_Hyp`: `While_Exact`, which directly expresses the semantics of loops in the postcondition.

$$\frac{}{\Gamma \vdash \{\!|\mathcal{P}|\!\}\ \textsf{while } B \textsf{ do } S \left\{\!\!\left|\begin{array}{l} \exists \sigma_{start} \in \mathcal{P}. b \in B. s \in S.\ \forall j.\ \exists l.\ \exists a_0, \cdots, a_l. \\ \sigma_{start}[j] = a_0 \wedge \\ \forall i, 0 \le i < l. [\![b]\!](a_i) = \textsf{true} \wedge [\![s]\!](a_i) = a_{i+1} \wedge \\ \sigma[j] = a_l \wedge [\![b]\!](a_l) = \textsf{false} \end{array}\right|\!\!\right\}} \text{While\_Exact}$$

We make the following remarks:

- `While_Exact` provides completeness by precisely capturing the semantics of loops directly in the postcondition. The explanation of `While_Exact` is as follows:
  - The postcondition of a set of while loops $\textsf{While } b \textsf{ do } s$ on a precondition of vector-states P can be precisely described as the set of vector-states σ for which the following holds.
  - There exists a loop $\textsf{While } b \textsf{ do } s$ with $b \in B$ and $s \in S$, such that for every individual j-th example in a precondition vector-state $\sigma_{start} \in P$, there exists a sequence of (individual) states $a_0, \cdots, a_l$, such that (i) a_0 is equivalent to the j-th example, (ii) $a_0, \cdots, a_l$ respect the semantics of the loop $\textsf{While } b \textsf{ do } s$, and (iii) the j-th example of the post-vector-state σ is equivalent to a_l.
- Because `While_Exact` computes the semantics of loops, it is true that it no longer supports invariant-based reasoning as in standard Hoare logic.

- However, invariant-based reasoning is still possible through `While_Hyp`. This rule suffices for many cases as shown in the paper; more precisely, one may still rely on `While_Hyp` if the desired property can be proved by using an invariant that holds over all possible loops in $\mathsf{While}\ B\ \mathsf{do}\ S$ (or, in tandem with the `GrmDisj` rule, by using finitely many invariants, each of which cover a subset that together span the entire space of possible loops).
- To construct a proof through invariant-based reasoning only, a user must be able to supply an invariant that is closely related to the precise semantics of the set of loops as described in `While_Exact` (in the sense that it is trivial to derive one from the other) anyways. Thus, one may argue that from a user perspective, the requirements for completeness have not changed.
- The postcondition of `While_Exact` is not an FO-PA formula in its current form, but there exists an equivalent FO-PA formulation through the Gödel β-function to encode the sequence of states, as well as the syntactic structure of terms, as a pair of existentially quantified integers.

Summary

In summary, we provide the following list of corrigenda to the paper, by section and page number.

- In Section 3.3, page 16, Figure 6, the single rule `While` should be replaced with the two rules `While_Hyp` and `While_Exact`.
- In Section 3.3, page 18, the paragraph *Loops* describing the original `While` rule should be updated with the explanations of `While_Hyp` and `While_Exact` provided in the corrigenda instead.
- In Section 4.1, page 20, the proof of Lemma 4.3 should be rectified by replacing the paragraph starting with "One thing to note about..." with the following paragraph.
    - The most interesting case in the proof of Lemma 4.3 is the case with loops; all other cases can be proved via structural induction. For loops, because unrealizability logic contains the rule `While_Exact`, which precisely captures the semantics of loops, any sound triple about a loop can be proved via `While_Exact` and an application of the `Weaken` rule.

The rest of the Section 4 is unaffected: this is because soundness (for hyperproperties) is recovered via `While_Hyp`, while the proof sketch for completeness relies on Lemma 4.3 (the proof of which is what this corrigenda mainly concerns).

Acknowledgements

We thank Shaan Nagy for discovering and helping to investigate the completeness bug.