Rely-Guarantee Specification of a Concurrent Garbage Collector

Roger C. Su and Nisansala P. Yatapanage

October 27, 2026

Abstract

We specify a model of the Ben-Ari concurrent garbage collector in the rely-guarantee paradigm, and verify its correctness. The aim is to show that the algorithm can be verified with rely-guarantee without the use of shared auxiliary variables. The approach is to reason that a certain property is maintained by one component for all instances of a variable, and then instantiate it to the particular instance required by the rely of the other component.

Contents

1 Introduction **1**

2 Preliminaries **2**

3 Definitions **3**

4 Lemmas **6**
4.1 Stability Lemmas 6
4.2 Lemma 1 7
4.3 Lemma 2 7

5 Theorems **7**
5.1 Holds for All 7
5.2 Propagate 8
5.3 Every Reachable Node Is Marked 9

1 Introduction

Jones and Yatapanage [JY19] showed that Ben-Ari's concurrent garbage collector [BA84] is difficult to specify in the rely-guarantee paradigm without shared auxiliary variables, which break compositionality. This proof-

development, which accompanies [Yat26a, Yat26b], expands on an alternative approach briefly mentioned in [JY19]. The idea is to maintain compositionality by using *Holds For All* reasoning. The Mutator is shown to satisfy a certain property over all instances of a variable, without requiring any knowledge about which particular instance the Collector requires for its rely condition. The property is then instantiated to the required context.

The specification focusses on the Collector's Marking phase, which is where the interference from the Mutator manifests. It specifies and establishes (1) that the Mutator maintains the property required for the Collector's rely to hold, and (2) that the postcondition of the Collector's Marking phase is reached.

Further details can be found in [Yat26a, Yat26b].

theory *RG-Garbage-Collector* **imports** *Main* **begin**

2 Preliminaries

lemma *holds-for-all*:
  assumes $guar\text{-}b\ \sigma\ \sigma'$
    and $p\ s\ \sigma$
    and $\forall\ w.\ (guar\text{-}b\ \sigma\ \sigma' \wedge p\ w\ \sigma \longrightarrow q\ w\ \sigma\ \sigma')$
    and $p\ s\ \sigma \wedge q\ s\ \sigma\ \sigma' \longrightarrow rely\text{-}a\ \sigma\ \sigma'$
  shows $rely\text{-}a\ \sigma\ \sigma'$
  ⟨*proof*⟩

definition *reach*
  $::\ 'a\ set \Rightarrow 'a\ rel \Rightarrow 'a\ set$ **where**
  $reach\ S\ R \equiv rtrancl\ R\ ``\ S$

lemma *reach-base*:
  $w \in S \Longrightarrow w \in reach\ S\ R$
  ⟨*proof*⟩

lemma *reach-step*:
  $[\![\ w \in reach\ S\ R\ ;\ (w,w') \in R\ ]\!] \Longrightarrow w' \in reach\ S\ R$
  ⟨*proof*⟩

lemma *reach-induct*:
  assumes $w \in reach\ S\ R$
    and $\bigwedge\ v.\ v \in S \Longrightarrow P\ v$
    and $\bigwedge\ u\ v.\ u \in reach\ S\ R \Longrightarrow P\ u \Longrightarrow (u,v) \in R \Longrightarrow P\ v$
  shows $P\ w$
  ⟨*proof*⟩

3 Definitions

The *nodes* in memory are represented as an arbitrary type $'a$.

Part of the system state is the local variables of the Collector, which cannot be accessed by the Mutator.

record $'a\ localC =$
  $safe :: 'a\ set$
  $consid :: 'a\ set$

The state of the system is defined as the following record-type.

record $'a\ state =$
  $roots :: 'a\ set$
  $hp :: 'a\ rel$ — the heap is modelled as a directed graph (i.e. a relation)
  $free :: 'a\ set$
  $marked :: 'a\ set$
  $tbm :: 'a\ option$
  $localCvar :: 'a\ localC$

Then we formulate the notion of "a node being reachable in a state".

definition *reachable*
  $:: 'a\ state \Rightarrow 'a \Rightarrow bool$ **where**
  $reachable\ \sigma\ x \equiv x \in reach\ (roots\ \sigma)\ (hp\ \sigma)$

lemma *reachable-exists*:
  $reachable\ \sigma\ y \Longrightarrow y \in roots\ \sigma \vee (\exists\ x.\ reachable\ \sigma\ x \wedge (x,y) \in hp\ \sigma)$
  $\langle proof \rangle$

We now define the various clauses that make up of our specification.

In the definitions below, if xy is a pair in the heap, then $fst\ xy$ is the node in the domain ($(a,j) \in dom\ hp$ in the paper), and $snd\ xy$ is any child of the node ($hp(a,j)$ in the paper).

All the marked nodes in the given set have marked children.

definition *all-ok*
  $:: 'a\ set \Rightarrow 'a\ state \Rightarrow bool$ **where**
  $all\text{-}ok\ s\ \sigma \equiv\ \forall\ xy \in hp\ \sigma.$
    $fst\ xy \in marked\ \sigma\ \wedge$
    $fst\ xy \in s$
    $\longrightarrow (snd\ xy \in marked\ \sigma)$

For all marked nodes in the given set that have unmarked children, there is another path to that child.

definition *can-be-marked*
  $:: 'a\ set \Rightarrow 'a\ state \Rightarrow bool$ **where**
  $can\text{-}be\text{-}marked\ s\ \sigma \equiv \forall\ xy \in hp\ \sigma.$
    $fst\ xy \in marked\ \sigma \cap s\ \wedge$

$snd\ xy \notin marked\ \sigma\ \wedge$
$reachable\ \sigma\ (snd\ xy)$
$\longrightarrow (\exists\ zw \in hp\ \sigma.$
$(fst\ zw \in\ reach\ (roots\ \sigma)\ (hp\ \sigma) \vee fst\ zw \in reach\ (marked\ \sigma \cap -s)\ (hp\ \sigma)) \wedge$
$(fst\ zw \notin s \vee fst\ zw \notin marked\ \sigma) \wedge$
$fst\ xy \neq fst\ zw \wedge$
$snd\ xy = snd\ zw)$

The *tbm* field is the only child node that (1) is unmarked and (2) has a marked parent.

definition *tbm-put*
$::\ 'a\ set \Rightarrow\ 'a\ state \Rightarrow bool$ **where**
$tbm\text{-}put\ s\ \sigma \equiv \forall\ xy \in hp\ \sigma.$
$tbm\ \sigma \neq Some\ (snd\ xy) \wedge$
$fst\ xy \in s \wedge$
$fst\ xy \in marked\ \sigma$
$\longrightarrow (snd\ xy \in marked\ \sigma)$

The rely condition of the collector.

definition *rely-collector*
$::\ 'a\ state \Rightarrow\ 'a\ state \Rightarrow bool$ **where**
$rely\text{-}collector\ \sigma\ \sigma' \equiv marked\ \sigma \subseteq marked\ \sigma' \wedge localCvar\ \sigma = localCvar\ \sigma' \wedge$
$(all\text{-}ok\ (safe\ (localCvar\ \sigma))\ \sigma \longrightarrow$
$can\text{-}be\text{-}marked\ (safe\ (localCvar\ \sigma))\ \sigma' \vee marked\ \sigma \subset marked\ \sigma')$

The guarantee condition of the Propagate phase of the Collector.

definition *guar-collector*
$::\ 'a\ state \Rightarrow\ 'a\ state\ \Rightarrow bool$ **where**
$guar\text{-}collector\ \sigma\ \sigma' \equiv marked\ \sigma \subseteq marked\ \sigma' \wedge$
$consid\ (localCvar\ \sigma) \subseteq consid\ (localCvar\ \sigma') \wedge$
$safe\ (localCvar\ \sigma') \subseteq safe\ (localCvar\ \sigma) \wedge$
$((safe\ (localCvar\ \sigma') = consid\ (localCvar\ \sigma')) \vee$
$((safe\ (localCvar\ \sigma') \subset consid\ (localCvar\ \sigma))) \wedge (marked\ \sigma \subset marked\ \sigma))$

The post condition of the Propagate phase of the Collector.

definition *post-propagate*
$::\ 'a\ state \Rightarrow\ 'a\ state\ \Rightarrow bool$ **where**
$post\text{-}propagate\ \sigma\ \sigma' \equiv \forall\ (x,y) \in hp\ \sigma.$
$x \in marked\ \sigma' \cap consid\ (localCvar\ \sigma') \longrightarrow$
$(y \in marked\ \sigma' \vee (marked\ \sigma \subset marked\ \sigma'))$

The post condition of the remove action of the Mutator.

definition *post-change-remove*
$::\ 'a\ state \Rightarrow\ 'a\ state \Rightarrow bool$ **where**
$post\text{-}change\text{-}remove\ \sigma\ \sigma' \equiv \exists\ xy.$
$xy \notin hp\ \sigma' \wedge hp\ \sigma = hp\ \sigma' \cup \{xy\} \wedge reachable\ \sigma\ (fst\ xy) \wedge$
$tbm\ \sigma' = None$

The post condition of the redirect action of the Mutator.

definition *post-change-redirect*
:: $'a$ *state* $\Rightarrow$ $'a$ *state* $\Rightarrow$ *bool* **where**
post-change-redirect σ σ' $\equiv$ $\exists$ x y z.
hp σ $-$ *hp* σ' $=$ $\{(x,y)\}$ $\wedge$ *hp* σ' $-$ *hp* σ $=$ $\{(x,z)\}$ $\wedge$ *reachable* σ x $\wedge$
tbm σ' $=$ *Some* z $\wedge$ *reachable* σ z

The common post condition of the remove and redirect actions of the Mutator. It describes the behaviour when the edge $(x,\ y)$ is removed or redirected to $(x,\ a)$.

definition *post-change*
:: $'a$ *state* $\Rightarrow$ $'a$ *state* $\Rightarrow$ *bool* **where**
post-change σ σ' $\equiv$ *roots* σ $\neq$ $\{\}$ $\wedge$
(*post-change-remove* σ σ' $\vee$ *post-change-redirect* σ σ') $\wedge$
(*marked* σ $=$ *marked* σ') $\wedge$
(*roots* σ $=$ *roots* σ') $\wedge$
(*tbm* σ $=$ *tbm* σ') $\wedge$
(*localCvar* σ $=$ *localCvar* σ')

The post condition of the action of the Mutator that sets *tbm*.

definition *post-mark-tbm*
:: $'a$ *state* $\Rightarrow$ $'a$ *state* $\Rightarrow$ *bool* **where**
post-mark-tbm σ σ' $\equiv$
((*tbm* σ $=$ *None* $\wedge$ *marked* σ' $=$ *marked* σ) $\vee$
($\exists x$. *tbm* σ $=$ *Some* x $\wedge$ *marked* σ' $=$ *marked* σ $\cup$ $\{x\}$)) $\wedge$
tbm σ' $=$ *None* $\wedge$ *hp* σ $=$ *hp* σ' $\wedge$ *roots* σ $=$ *roots* σ' $\wedge$
(*localCvar* σ $=$ *localCvar* σ')

An invariant that the roots always remain marked.

definition *inv-roots*
:: $'a$ *state* $\Rightarrow$ *bool* **where**
inv-roots σ $\equiv$ *roots* σ $\subseteq$ *marked* σ

lemma *inv-roots-change*:
$[\![$ *post-change* σ σ' ; *inv-roots* σ $]\!]$ $\Longrightarrow$ *inv-roots* σ'
$\langle$*proof*$\rangle$

lemma *inv-roots-mark*:
$[\![$ *post-mark-tbm* σ σ' ; *inv-roots* σ $]\!]$ $\Longrightarrow$ *inv-roots* σ'
$\langle$*proof*$\rangle$

A Mutator step preserves the invariant.

lemma *inv-roots-step*:
$[\![$ *post-change* σ σ' $\vee$ *post-mark-tbm* σ σ' ; *inv-roots* σ $]\!]$ $\Longrightarrow$ *inv-roots* σ'
$\langle$*proof*$\rangle$

4 Lemmas

This lemma establishes one of the premises when instantiating *holds-for-all* to the concurrent garbage collector.

lemma *p-q-gives-rely*:
  assumes *all-ok* (*safe* (*localCvar* σ)) σ
    and *marked* $\sigma \subseteq$ *marked* $\sigma' \wedge$ *localCvar* $\sigma =$ *localCvar* $\sigma' \wedge$
      (*can-be-marked* (*safe* (*localCvar* σ)) $\sigma' \vee$ *marked* $\sigma \subset$ *marked* σ')
  shows *rely-collector* σ σ'
  ⟨*proof*⟩

lemma *all-ok-conseqs*:
  assumes *all-ok* s σ
  shows *can-be-marked* s σ
    and *tbm-put* s σ
  ⟨*proof*⟩

4.1 Stability Lemmas

A Mutator remove step preserves *all-ok*.

lemma *stable-all-ok-remove*:
  assumes *post-change* σ $\sigma' \wedge$ *post-change-remove* σ σ'
    and *all-ok* s σ
  shows *all-ok* s σ'
  ⟨*proof*⟩

lemma *reachable-post-pre*:
  assumes *assm-step*: *post-change* σ σ'
    and *assm-ok*: *all-ok* s σ
  shows $\forall$ x. *reachable* σ' $x \longrightarrow$ *reachable* σ x
⟨*proof*⟩

The *all-ok* property and a redirect step by the Mutator together establishes *can-be-marked*.

lemma *stable-can-be-marked-redirect*:
  assumes *assm-step*: *post-change* σ $\sigma' \wedge$ *post-change-redirect* σ σ'
    and *assm-ok*: *all-ok* s σ
    and *assm-roots-marked*: *roots* $\sigma \subseteq$ *marked* σ
  shows *can-be-marked* s σ'
⟨*proof*⟩

lemma *stable-can-be-marked-remove*:
  assumes *post-change* σ $\sigma' \wedge$ *post-change-remove* σ σ'
    and *all-ok* s σ
  shows *can-be-marked* s σ'
  ⟨*proof*⟩

4.2 Lemma 1

This subsection establishes Lemma 1 from the paper, which concerns the properties that follow from a Mutator change.

lemma *lemma-1-can-be-marked*:
  assumes *assm-step*: *post-change* σ σ'
    and *assm-all-ok*: *all-ok* s σ
    and *assm-roots-marked*: *roots* $\sigma \subseteq$ *marked* σ
  shows *can-be-marked* s σ'
⟨*proof*⟩

lemma *lemma-1-tbm-put*:
  assumes *assm-step*: *post-change* σ σ'
    and *assm-all-ok*: *all-ok* s σ
  shows *tbm-put* s σ'
⟨*proof*⟩

lemma *lemma-1*:
  assumes *post-change* σ σ'
    and *all-ok* s σ
    and *roots* $\sigma \subseteq$ *marked* σ
  shows *can-be-marked* s $\sigma' \wedge$ *tbm-put* s σ'
  ⟨*proof*⟩

4.3 Lemma 2

This subsection establishes Lemma 2 from the paper: after a Mutator step that marks *tbm* and *tbm-put*, either *all-ok* holds or the number of marked nodes increases.

lemma *lemma-2*:
  assumes *tbm-put* s σ
    and *post-mark-tbm* σ σ'
  shows *all-ok* s $\sigma' \vee$ (*marked* $\sigma \subset$ *marked* σ')
  ⟨*proof*⟩

5 Theorems

5.1 Holds for All

This establishes the *forall-guar-p-w* premise when instantiating the *holds-for-all* lemma to the concurrent garbage collector. Here, p is *all-ok* w σ and q is *can-be-marked* w $\sigma' \vee$ *marked* $\sigma \subset$ *marked* σ'.

lemma *works-for-all-w*:
  assumes *post-change* σ $\sigma' \vee$ *post-mark-tbm* σ σ'
    and *inv-roots* σ
    and *all-ok* w σ
  shows *can-be-marked* w $\sigma' \vee$ *marked* $\sigma \subset$ *marked* σ'

⟨*proof*⟩

This lemma continues the *works-for-all-w* reasoning, but with q extended to include the case where the Collector's local variables are unchanged and the marking only increases.

theorem *works-for-all-w2*:
assumes *post-change* σ σ' $\vee$ *post-mark-tbm* σ σ'
and *inv-roots* σ
and *all-ok* w σ
shows *marked* $\sigma \subseteq$ *marked* $\sigma' \wedge$ *localCvar* $\sigma =$ *localCvar* $\sigma' \wedge$
(*can-be-marked* w $\sigma' \vee$ *marked* $\sigma \subset$ *marked* σ')
⟨*proof*⟩

This theorem shows that the rely of the Collector holds, based on the *holds-for-all* reasoning.

theorem *using-holds-for-all2*:
assumes *post-change* σ σ' $\vee$ *post-mark-tbm* σ σ'
and *all-ok* s σ
and $\forall$ w. ((*post-change* σ σ' $\vee$ *post-mark-tbm* σ σ') $\wedge$ *all-ok* w σ)
$\longrightarrow$ (*marked* $\sigma \subseteq$ *marked* $\sigma' \wedge$ *localCvar* $\sigma =$ *localCvar* $\sigma' \wedge$
(*can-be-marked* w $\sigma' \vee$ *marked* $\sigma \subset$ *marked* σ'))
and *all-ok* s $\sigma \wedge$ *marked* $\sigma \subseteq$ *marked* $\sigma' \wedge$ *localCvar* $\sigma =$ *localCvar* $\sigma' \wedge$
(*can-be-marked* s $\sigma' \vee$ *marked* $\sigma \subset$ *marked* σ')
$\longrightarrow$ *rely-collector* σ σ'
shows *rely-collector* σ σ'
⟨*proof*⟩

5.2 Propagate

lemma *chain-mono*:
$[\![\bigwedge m.\ f\ m \subseteq f\ (Suc\ m)\ ;\ a \leq b\ ]\!] \Longrightarrow f\ a \subseteq f\ b$
⟨*proof*⟩

lemma *chain-anti*:
$[\![\bigwedge m.\ f\ (Suc\ m) \subseteq f\ m\ ;\ a \leq b\ ]\!] \Longrightarrow f\ b \subseteq f\ a$
⟨*proof*⟩

This theorem shows that *post-propagate* holds over a sequence of steps, where each step either satisfies *rely-collector* or *guar-collector*, and the *safe* and *consid* sets are initially empty.

theorem *propagate-seq*:
fixes *st* :: *nat* $\Rightarrow$ *'a state*
assumes *step*: $\bigwedge$ m. *rely-collector* (*st* m) (*st* (*Suc* m))
$\vee$ *guar-collector* (*st* m) (*st* (*Suc* m))
and *init-empty*: *safe* (*localCvar* (*st* 0)) = {} $\wedge$
consid (*localCvar* (*st* 0)) = {}
shows *post-propagate* (*st* 0) (*st* n)
⟨*proof*⟩

5.3 Every Reachable Node Is Marked

This final theorem establishes the overall post condition of the Collector, that "if a node is reachable initially, then it has been marked by the Collector.

theorem *reachable-subset-marked*:
assumes *assm-post*: *post-propagate* σ σ'
and *assm-marked*: *marked* $\sigma' =$ *marked* σ
and *assm-roots*: *roots* $\sigma \subseteq$ *marked* σ
and *assm-consid*: *consid* (*localCvar* σ') = *Field* (*hp* σ)
— *consid* contains all the nodes in the heap.
shows $\forall$ *w*. *reachable* σ *w* $\longrightarrow$ *w* $\in$ *marked* σ
⟨*proof*⟩

end

References

[BA84] Mordechai Ben-Ari. Algorithms for on-the-fly garbage collection. *ACM Transactions on Programming Languages and Systems*, 6(3):333–344, 1984. doi:10.1145/579.587.

[JY19] Cliff B. Jones and Nisansala Yatapanage. Investigating the limits of rely/guarantee relations based on a concurrent garbage collector example. *Formal Aspects of Computing*, 31:353–374, 2019. doi:10.1007/s00165-019-00482-3.

[Yat26a] Nisansala P. Yatapanage. The complexity of interference: When rely/guarantee does not work. *CoRR*, abs/2609.27392, 2026. arXiv:2609.27392.

[Yat26b] Nisansala P. Yatapanage. The complexity of interference: When rely/guarantee does not work. In *The 12th International Symposium on Dependable Software Engineering: Theories, Tools and Applications (SETTA)*, 2026. to appear.