Understanding Lua's Garbage Collection

Towards a Formalized Static Analyzer

Mallku Soldevila
FAMAF, UNC and CONICET
Argentina
mes0107@famaf.unc.edu.ar

Beta Ziliani
FAMAF, UNC and CONICET
Argentina
beta@mpi-sws.org

Daniel Fridlender
FAMAF, UNC
Argentina
fridlend@famaf.unc.edu.ar

Appendices

A PROPERTIES OF GC

To reach to a proof of the correctness of $\overset{\text{GC}}{\mapsto}$ we will require, first, to check for several lemmas about simple properties that hold for both, $\overset{\text{L}}{\mapsto}$ and $\overset{\text{GC}}{\mapsto}$.

Properties preserved by $\overset{\text{L}}{\mapsto}$. The first lemma states that once a binding becomes amenable for collection, it will remain in that state after any computation step from $\overset{\text{L}}{\mapsto}$.[1] For its proof we will assume that the reader is familiar with the model presented in [?]. A complete proof would require case analysis on every computation step from said model. For reasons of brevity, we will consider just a few cases.

Lemma A.1. *For configurations* $(\sigma_1 : \theta_1 : s_1)$, $(\sigma_2 : \theta_2 : s_2)$*, if* $(\sigma_1 : \theta_1 : s_1) \overset{\text{L}}{\mapsto} (\sigma_2 : \theta_2 : s_2)$*, for* $(\sigma_1 : \theta_1 : s_1)$ *well-formed, then* $\forall l \in dom(\sigma_1) \cup dom(\theta_1)$, $\neg\text{reach}(l, s_1, \sigma_1, \theta_1) \Rightarrow \neg\text{reach}(l, s_2, \sigma_2, \theta_2)$.

Proof. We will follow the modular structure of $\overset{\text{L}}{\mapsto}$ to reason over the step that transforms $(\sigma_1 : \theta_1 : s_1)$ into $(\sigma_2 : \theta_2 : s_2)$. We have the following cases for the step taken from $\overset{\text{L}}{\mapsto}$:

- *The computation does not depend on the content of the stores (*i.e.*, it does not change bindings from a store or dereferences locations):* then, it can be seen, by case analysis on each computation rule, that such computation step does not introduce any reference into the instruction term. What could happen is that the root set is reduced, by deleting references present into s_1. In any case, for a given $l \in dom(\sigma_1) \cup dom(\theta_1)$, if $\neg\text{reach}(l, s_1, \sigma_1, \theta_1)$ it must be the case that also $\neg\text{reach}(l, s_2, \sigma_2, \theta_2)$.
- *The computation changes or dereferences locations from* σ_1 : for a given $l \in dom(\sigma_1) \cup dom(\theta_1)$, such that $\neg\text{reach}(l, s_1, \sigma_1, \theta_1)$ let us assume that $\text{reach}(l, s_2, \sigma_2, \theta_2)$. To reason about the statement, we would need to do case analysis on every possible computation step that interacts with the values store. As an example, let us consider the implicit dereferencing of references to σ_1.[2] Then it must be the case that s_1 matches against the pattern $E[\![\ r\ ]\!]$, for an evaluation context E and a reference r, and the computation is:

$$\sigma_1 : \theta_1 : \overset{\underline{\underline{s_1}}}{E[\![\ r\ ]\!]} \overset{\text{L}}{\mapsto} \sigma_1 : \theta_1 : \overset{\underline{\underline{s_2}}}{E[\![\sigma_1(r)]\!]}$$

where both stores remain unmodified after the computation. Then, the root set just changed by replacing r by the references in $\sigma_1(r)$. If $\neg\text{reach}(l, s_1, \sigma_1, \theta_1)$ but $\text{reach}(l, s_2, \sigma_2, \theta_2)$, this would mean that l is reachable from the references in $\sigma_1(r)$. But in s_1, the references from $\sigma_1(r)$ were also reachable, making l reachable in s_1, contradicting our hypothesis. Then, it must be the case that if $\neg\text{reach}(l, s_1, \sigma_1, \theta_1)$, l remains unreachable in $(\sigma_2 : \theta_2 : s_2)$.
- *The computation changes or dereferences locations from* θ_1: let us assume that for a given $l \in dom(\sigma_1) \cup dom(\theta_1)$, $\neg\text{reach}(l, s_1, \sigma_1, \theta_1) \ \wedge\ \text{reach}(l, s_2, \sigma_2, \theta_2)$. Again we will just analyze one case, among every computation that interacts with the store θ_1. We will consider the rule that describes how tables are allocated in θ_1. Then, it must be the case that s_1 matches against the pattern $E[\![\ t\ ]\!]$, for an evaluation context E and a table constructor t, where every field haven been evaluated, making the table ready for allocation. Then, the (simplified) computation is:

$$\frac{tid \notin \text{dom}(\theta_1) \qquad \theta_2 = (tid,\ (t,\ \textbf{nil},\ \bot)),\ \theta_1}{(\sigma_1\ :\ \theta_1\ :\ E[\![\ t\ ]\!])\ \overset{\text{L}}{\mapsto}\ (\sigma_1\ :\ \theta_2\ :\ E[\![\ tid\ ]\!])}$$

where the values store remains unchanged, *i.e.*, $\sigma_2 = \sigma_1$. Then, the root set just changed by replacing the references in t by the fresh table identifier tid. If $\neg\text{reach}(l, s_1, \sigma_1, \theta_1)$ but $\text{reach}(l, s_2, \sigma_2, \theta_2)$, this would mean that $l = tid$, which cannot be the case as $l \in dom(\sigma_1) \cup dom(\theta_1)$ and $tid \notin dom(\sigma_1) \cup dom(\theta_1)$. Then it must be the case that if $\neg\text{reach}(l, s_1, \sigma_1, \theta_1)$, l remains unreachable in $(\sigma_2 : \theta_2 : s_2)$.

□

The following definition and lemma capture a standard concept in operational semantics for imperative languages: for a given instruction term, the outcome of its execution under given stores will depend on the content of the reachable portion of said stores.

Definition A.2. For well-formed configurations $(\sigma_1 : \theta_1 : s)$ and $(\sigma_2 : \theta_2 : s)$, we will say that both configurations *coincide in the reachable portion of their stores*, denoted

$$(\sigma_1 : \theta_1 : s) \overset{\text{rch}}{\sim} (\sigma_2 : \theta_2 : s)$$

if and only if $\forall l \in dom(\sigma_1) \cup dom(\theta_1) / \text{reach}(l, s, \sigma_1, \theta_1)$, then:

- $\text{reach}(l, s, \sigma_2, \theta_2)$
- $l \in dom(\sigma_1) \Rightarrow \sigma_1(l) = \sigma_2(l)$
- $l \in dom(\theta_1) \Rightarrow \theta_1(l) = \theta_2(l)$

and the same holds $\forall l \in dom(\sigma_2) \cup dom(\theta_2)$.

[1] Note that such simple property does not hold anymore if we introduce weak tables or finalization.
[2] In [?], for purposes of simplification of the desugared Lua code from test suites, we included implicit dereferencing of references to values, as done in [?].

Conference'17, July 2017, Washington, DC, USA
2020. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

In the previous definition, we are assuming that, if needed, it is always possible to provide a renaming of locations from both configurations to make them equivalent in the sense expressed by $\overset{\mathsf{rch}}{\sim}$. Finally, it is easy to show that $\overset{\mathsf{rch}}{\sim}$ is an equivalence relation.

The important property, satisfied by configurations that coincide in the reachable portion of their stores, is stated in the following lemmas:

Lemma A.3. *For well-formed configurations* $(\sigma_1 : \theta_1 : s_1)$ *and* $(\sigma_2 : \theta_2 : s_1)$*, such that:*

$$(\sigma_1 : \theta_1 : s_1) \overset{\mathsf{rch}}{\sim} (\sigma_2 : \theta_2 : s_1)$$

if $\exists(\sigma_3 : \theta_3 : s_2)/(\sigma_1 : \theta_1 : s_1) \overset{\mathsf{L}}{\mapsto} (\sigma_3 : \theta_3 : s_2)$*, then* $\exists(\sigma_4 : \theta_4 : s_2)/(\sigma_2 : \theta_2 : s_1) \overset{\mathsf{L}}{\mapsto} (\sigma_4 : \theta_4 : s_2)$ *and:*

$$(\sigma_3 : \theta_3 : s_2) \overset{\mathsf{rch}}{\sim} (\sigma_4 : \theta_4 : s_2)$$

Proof. We will follow the modular structure of $\overset{\mathsf{L}}{\mapsto}$ to reason over the step that transforms $(\sigma_2 : \theta_2 : s_1)$ into $(\sigma_4 : \theta_4 : s_2)$:

- *The computation does not change bindings from a store or dereferences locations* : then it must be the case that every information from the stores is already put into the instruction term s_1 so as to make the computation from $\overset{\mathsf{L}}{\mapsto}$ viable, without regard to the content of the stores. Also, after the computation the stores are not modified. It implies that:

$$(\sigma_1 : \theta_1 : s_1) \overset{\mathsf{L}}{\mapsto} (\overset{\sigma_1}{\overset{=}{\sigma_3}} : \overset{\theta_1}{\overset{=}{\theta_3}} : s_2) \ \wedge\ (\sigma_2 : \theta_2 : s_1) \overset{\mathsf{L}}{\mapsto} (\overset{\sigma_2}{\overset{=}{\sigma_4}} : \overset{\theta_2}{\overset{=}{\theta_4}} : s_2)$$

  The root set of references in both configurations, $(\sigma_3 : \theta_3 : s_2)$ and $(\sigma_4 : \theta_4 : s_2)$, is the same. And, since $(\sigma_1 : \theta_1 : s_1) \overset{\mathsf{rch}}{\sim} (\sigma_2 : \theta_2 : s_1)$ and the stores are not modified after the step from $\overset{\mathsf{L}}{\mapsto}$, it follows that the reachable portion of the stores, from the root set defined by s_2, must coincide, according to $\overset{\mathsf{rch}}{\sim}$, in the configurations obtained after $\overset{\mathsf{L}}{\mapsto}$. Hence:

$$(\sigma_3 : \theta_3 : s_2) \overset{\mathsf{rch}}{\sim} (\sigma_4 : \theta_4 : s_2)$$

- *The computation changes or dereferences locations from* σ_1: we would need to do case analysis on each computation that interacts with the value store. As an example, let us consider the implicit dereferencing of references to the values store. The hypothesis can be rewritten as:

$$(\sigma_1 : \theta_1 : s_1) \overset{\mathsf{L}}{\mapsto} (\overset{\sigma_1}{\overset{=}{\sigma_3}} : \overset{\theta_1}{\overset{=}{\theta_3}} : s_2)$$

  where s_2 contains the value associated with the reference dereferenced by $\overset{\mathsf{L}}{\mapsto}$. Because:

$$(\sigma_1 : \theta_1 : s_1) \overset{\mathsf{rch}}{\sim} (\sigma_2 : \theta_2 : s_1)$$

  the dereferencing operation will return the same result, if executed over σ_2. Then:

$$(\sigma_2 : \theta_2 : s_1) \overset{\mathsf{L}}{\mapsto} (\overset{\sigma_2}{\overset{=}{\sigma_4}} : \overset{\theta_2}{\overset{=}{\theta_4}} : s_2)$$

  Finally, because the stores are unmodified, after the step from $\overset{\mathsf{L}}{\mapsto}$, and since the reachable portions of the stores in the original configurations coincide, according to $\overset{\mathsf{rch}}{\sim}$, then, it must be the case that the reachable portions of the stores obtained after $\overset{\mathsf{L}}{\mapsto}$ must also coincide, if we consider the same root of references. Hence:

$$(\sigma_3 : \theta_3 : s_2) \overset{\mathsf{rch}}{\sim} (\sigma_4 : \theta_4 : s_2)$$

- *The computation changes or dereferences locations from* θ_1: we would need to do case analysis on each computation that interacts with θ_1. As an example, let us consider table allocation. The hypothesis can be rewritten as:

$$(\sigma_1 : \theta_1 : E[\![t]\!]) \overset{\mathsf{L}}{\mapsto} (\overset{\sigma_3}{\overset{=}{\sigma_1}} : \overset{\theta_3}{\overset{=}{\theta_1 \uplus \{(tid, t, \mathbf{nil}, \bot)\}}} : E[\![tid]\!])$$

  Then we can assume that:

$$(\sigma_2 : \theta_2 : E[\![t]\!]) \overset{\mathsf{L}}{\mapsto} (\overset{\sigma_4}{\overset{=}{\sigma_2}} : \overset{\theta_4}{\overset{=}{\theta_2 \uplus \{(tid, t, \mathbf{nil}, \bot)\}}} : E[\![tid]\!])$$

  where, if needed, we could apply a consistent renaming of tables' id in $(\sigma_2 : \theta_2 : E[\![t]\!])$, such that it preserves its equivalence with $(\sigma_1 : \theta_1 : E[\![t]\!])$ and *tid* is available as a fresh table identifier. Then, it follows immediately that:

$$(\sigma_3 : \theta_3 : E[\![tid]\!]) \overset{\mathsf{rch}}{\sim} (\sigma_4 : \theta_4 : E[\![tid]\!])$$

□

Finally, the following lemma express an intuitive property that holds among final configurations that happen to be equivalent, according to $\overset{\mathsf{rch}}{\sim}$:

Lemma A.4. *For final configurations* $(\sigma_1 : \theta_1 : s)$ *and* $(\sigma_2 : \theta_2 : s)$*, such that* $(\sigma_1 : \theta_1 : s) \overset{\mathsf{rch}}{\sim} (\sigma_2 : \theta_2 : s)$*, then:*

$$\mathsf{result}(\sigma_1 : \theta_1 : s) = \mathsf{result}(\sigma_2 : \theta_2 : s)$$

Proof. The result will follow directly from the definition of result, in Figure 14, and $\overset{\mathsf{rch}}{\sim}$. We will do a case analysis on the structure of s, for the configuration $(\sigma_1 : \theta_1 : s)$, considering that it is the final state of a convergent computation:

- $s = \mathbf{return} v_1, ..., v_n$: for simplicity we consider the case $n = 1$, and we omit a possible context E where the **return** statement could occur, since it is not taken into account by the notion of result of a program, as defined by result. For larger values of n the reasoning remains the same:
  - $v_1 \in number \cup string$: then, neither $\mathsf{result}(\sigma_1 : \theta_1 : s)$ nor $\mathsf{result}(\sigma_2 : \theta_2 : s)$ depend on the content of the stores. Hence, $\mathsf{result}(\sigma_1 : \theta_1 : s) = \mathsf{result}(\sigma_2 : \theta_2 : s)$.
  - $v_1 \in tid \cup cid$: let us consider that $v_1 = tid$ for some $tid \in \mathrm{dom}(\theta_1)$ (the reasoning for the case $v_1 \in cid$ is similar). Then, by definition of result:
    $\mathsf{result}(\sigma_1 : \theta_1 : \mathbf{return}\ tid) = \sigma_1|_{\mathsf{S}} : \theta_1|_{\mathsf{T}} : \mathbf{return}\ tid$

    $$\text{where} \begin{cases} \mathsf{S} = \bigcup\limits_{r \in \mathrm{dom}(\sigma_1),\ \mathsf{reach}(r, \mathbf{return}\ tid, \sigma_1, \theta_1)} r \\ \\ \mathsf{T} = \bigcup\limits_{id \in \mathrm{dom}(\theta_1),\ \mathsf{reach}(id, \mathbf{return}\ tid, \sigma_1, \theta_1)} id \end{cases}$$

    Then, clearly $\mathsf{result}(\sigma_1 : \theta_1 : \mathbf{return}\ tid)$ is depending on the reachable portions of σ_1 and θ_1, beginning with the root set defined by *tid*. Because

$$(\sigma_1 : \theta_1 : tid) \overset{\mathsf{rch}}{\sim} (\sigma_2 : \theta_2 : tid)$$

    the reachable portions of both configurations coincide. Hence, $\mathsf{result}(\sigma_1 : \theta_1 : s) = \mathsf{result}(\sigma_2 : \theta_2 : s)$.
  - $s = \mathbf{error}\ v$: this case is identical to the previous one.

- s = ;: then, result($\sigma_1 : \theta_1 : s$) is not depending on the content of the stores, and so is the case for result($\sigma_2 : \theta_2 : s$). Hence,
$$\text{result}(\sigma_1 : \theta_1 : s) = \text{result}(\sigma_2 : \theta_2 : s)$$

□

Properties preserved by $\overset{\text{GC}}{\mapsto}$. A simple property to ask for is that reachable bindings are preserved, in the sense that they are still reachable and the value to which a given location is mapped is not changed after $\overset{\text{GC}}{\mapsto}$. We express this property with the following two lemmas:

Lemma A.5. *For a well-formed configuration* ($\sigma_1 : \theta_1 : s$)*, if* ($\sigma_1 : \theta_1 : s$) $\overset{\text{GC}}{\mapsto}$ ($\sigma_2 : \theta_2 : s$)*, for some configuration* ($\sigma_2 : \theta_2 : s$)*, then* $\forall r \in dom(\sigma_1)$, reach$(r, s, \sigma_1, \theta_1) \Rightarrow \sigma_1(r) = \sigma_2(r)$. *The analogous holds for any* $id \in dom(\theta_1)$.

Proof. Let $r \in dom(\sigma_1)$, reach$(r, s, \sigma_1, \theta_1)$. Then, by Definition 3.2 and $\overset{\text{GC}}{\mapsto}$, it must be the case that gc$(s, \sigma_1, \theta_1) = (\sigma_2, \theta_2)$ and $\sigma_1(r) = \sigma_2(r)$.

For elements from dom(θ_1) the reasoning is analogous to the previous case. □

Lemma A.6. *For a well-formed configuration* ($\sigma_1 : \theta_1 : s$)*, if* ($\sigma_1 : \theta_1 : s$) $\overset{\text{GC}}{\mapsto}$ ($\sigma_2 : \theta_2 : s$)*, for some configuration* ($\sigma_2 : \theta_2 : s$)*, then* $\forall l \in dom(\sigma_1) \cup dom(\theta_1)$, reach$(l, s, \sigma_1, \theta_1) \Rightarrow$ reach$(l, s, \sigma_2, \theta_2)$.

Proof. We will prove it by induction on the minimum number of dereferences of locations from σ_1 or θ_1 that needs to be performed to reach to a given location l, for which reach$(l, s, \sigma_1, \theta_1)$ holds. By looking at Definition 3.1, one of the following cases should hold:

- $l \in s$: then it follows directly that reach$(l, s, \sigma_2, \theta_2)$.
- $\exists r \in$ dom(σ_1), $l \in \sigma_1(r)$, which is in a reachability path of minimum distance, from the root set to r : then reach$(r, s, \sigma_1, \theta_1)$, and by inductive hypothesis, reach$(r, s, \sigma_2, \theta_2)$. Also, by lemma A.5, $\sigma_1(r) = \sigma_2(r)$. Then $l \in \sigma_2(r)$ and reach$(l, s, \sigma_2, \theta_2)$, by definition.
- $\exists tid \in$ dom(θ_1), $l \in \pi_1(\theta_1(tid))$, which is in a reachability path of minimum distance, from the root set to l: then reach$(tid, s, \sigma_1, \theta_1)$, and by inductive hypothesis, reach$(tid, s, \sigma_2, \theta_2)$. Also, by Lemma A.5, $\theta_1(tid) = \theta_2(tid)$. Then $l \in \pi_1(\theta_2(tid))$ and reach$(l, s, \sigma_2, \theta_2)$ by definition.
- $\exists cid \in$ dom(θ_1), $l \in \theta_1(cid)$, which is in a reachability path of minimum distance, from the root set to l: the reasoning is analogous to the previous case. It follows directly that reach$(l, s, \sigma_2, \theta_2)$.
- $\exists tid \in$ dom(θ_1), $l \in \pi_2(\theta_2(tid))$, which is in a reachability path of minimum distance, from the root set to l: the situation is analogous to the previous case. It follows directly that reach$(l, s, \sigma_2, \theta_2)$.

□

Corollary A.7. *For well-formed configurations* ($\sigma_1 : \theta_1 : s$) *and* ($\sigma_2 : \theta_2 : s$)*, if* ($\sigma_1 : \theta_1 : s$) $\overset{\text{GC}}{\mapsto}$ ($\sigma_2 : \theta_2 : s$)*, then* ($\sigma_1 : \theta_1 : s$) $\overset{\text{rch}}{\sim}$ ($\sigma_2 : \theta_2 : s$).

Proof. It is a direct consequence of lemmas A.5, A.6 and the definition of $\overset{\text{rch}}{\sim}$. □

While the following lemma directly refers to the notion of well-formedness of configurations, it is not required to describe in detail such notion in order to gain confidence about the following statement and its proof, since they are intuitive enough (for details about well-formedness, we refer the reader to [?]). Also, the lemma will allow us to extend the mentioned progress property for $\overset{\text{L}}{\mapsto}$ to the semantics obtained adding $\overset{\text{GC}}{\mapsto}$. In particular, it will guarantee that the introduced notion of observations over programs is well-defined also for $\overset{\text{L}}{\mapsto} \cup \overset{\text{GC}}{\mapsto}$, allowing us to state the desired correctness for $\overset{\text{GC}}{\mapsto}$.

Lemma A.8. *For a well-formed configuration* ($\sigma_1 : \theta_1 : s$)*, if* ($\sigma_1 : \theta_1 : s$) $\overset{\text{GC}}{\mapsto}$ ($\sigma_2 : \theta_2 : s$)*, for some configuration* ($\sigma_2 : \theta_2 : s$)*, then* ($\sigma_2 : \theta_2 : s$) *is well-formed.*

Proof. From the definition of $\overset{\text{GC}}{\mapsto}$, it follows that the step does not change the instruction term. Also, by the previous lemmas, it follows that $\overset{\text{GC}}{\mapsto}$ does not introduce dangling pointers. They also state that $\overset{\text{GC}}{\mapsto}$ does not modify the stores in any other way, besides removing garbage. Then, it must be the case that also ($\sigma_2 : \theta_2 : s$) is well-formed. □

Lemma A.9. *Over a well-formed configuration* ($\sigma : \theta : s$)*, only a finite number of* $\overset{\text{GC}}{\mapsto}$ *steps can be applied.*

Proof. By Definition 3.2 and $\overset{\text{GC}}{\mapsto}$, if
$$(\sigma : \theta : s) \overset{\text{GC}}{\mapsto} (\sigma' : \theta' : s)$$
then it must be the case that either σ' or θ' is a proper subset of σ or θ, respectively. Then, being the stores partial finite functions, it is clear that GC can be performed at most a finite number of steps. □

The following lemma is a useful tool taken from [?]. It codifies a simple intuition of plain GC: it must be possible to postpone any GC step, without changing the observations of the program. In its statement we use the fact that $\overset{\text{GC}}{\mapsto}$ does not change the instruction term.

Lemma A.10 (Postponement). *For a given well-formed configuration* ($\sigma_1 : \theta_1 : s_1$)*, if*
$$(\sigma_1 : \theta_1 : s_1) \overset{\text{GC}}{\mapsto} (\sigma_2 : \theta_2 : s_1) \overset{\text{L}}{\mapsto} (\sigma_3 : \theta_3 : s_2).$$
then $\exists(\sigma_4 : \theta_4 : s_2)$ *such that:*
$$(\sigma_1 : \theta_1 : s_1) \overset{\text{L}}{\mapsto} (\sigma_4 : \theta_4 : s_2) \overset{\text{GC}}{\mapsto} (\sigma_3' : \theta_3' : s_2)$$
where ($\sigma_3 : \theta_3 : s_2$) $\overset{\text{rch}}{\sim}$ ($\sigma_3' : \theta_3' : s_2$).

Proof. We will follow the modular structure of $\overset{\text{L}}{\mapsto}$ to reason over the step that transforms ($\sigma_2 : \theta_2 : s_1$) into ($\sigma_3 : \theta_3 : s_2$):

- *The computation does not change bindings from a store or dereferences locations*: then it must be the case that every information from the stores is already put into the instruction term s_1 so as to make the computation from $\overset{\text{L}}{\mapsto}$ viable, without regard to the content of the stores. Then, the hypothesis can be rewritten as:
$$(\sigma_1 : \theta_1 : s_1) \overset{\text{GC}}{\mapsto} (\sigma_2 : \theta_2 : s_1) \overset{\text{L}}{\mapsto} (\sigma_2 : \theta_2 : s_2)$$

If we take $(\sigma_4 : \theta_4 : s_2) = (\sigma_1 : \theta_1 : s_2)$, then we can assert that:

$$(\sigma_1 : \theta_1 : s_1) \overset{\mathsf{L}}{\mapsto} (\sigma_1 : \theta_1 : s_2) = (\sigma_4 : \theta_4 : s_2)$$

where we exploited the fact that, for the previous $\overset{\mathsf{L}}{\mapsto}$ step to be performed, the actual content of the stores does not affect the applicability and the outcome of said computation. Then, by Lemma A.1, if a binding was ready to be collected in $(\sigma_1 : \theta_1 : s_1)$ it will remain in that state in $(\sigma_1 : \theta_1 : s_2)$. So, by the non-deterministic nature of $\overset{\mathsf{GC}}{\mapsto}$, we could ask for it to remove the same bindings that changed the stores from $(\sigma_1 : \theta_1 : s_1)$ into the stores from $(\sigma_2 : \theta_2 : s_1)$. Hence, it must be the case that $(\sigma_1 : \theta_1 : s_2) \overset{\mathsf{GC}}{\mapsto} (\sigma_2 : \theta_2 : s_2)$ holds. We obtained:

$$(\sigma_1 : \theta_1 : s_1) \overset{\mathsf{L}}{\mapsto} (\sigma_1 : \theta_1 : s_2) \overset{\mathsf{GC}}{\mapsto} (\sigma_2 : \theta_2 : s_2)$$

Finally, $(\sigma_3 : \theta_3 : s_2) \overset{\mathsf{rch}}{\sim} (\sigma_3' : \theta_3' : s_2)$ because

$$(\sigma_3 : \theta_3 : s_2) = (\sigma_2 : \theta_2 : s_2) = (\sigma_3' : \theta_3' : s_2)$$

- *The computation changes or dereferences locations from σ_1*: we would need to do case analysis on each computation that interacts with the value store. As an example, let us consider the implicit dereferencing of a reference to σ_1. That is, the $\overset{\mathsf{L}}{\mapsto}$ step should be:

$$(\sigma_2 : \theta_2 : \overset{\overset{s_1}{\equiv}}{E[\![r]\!]}) \overset{\mathsf{L}}{\mapsto} (\sigma_2 : \theta_2 : \overset{\overset{s_2}{\equiv}}{E[\![\sigma_2(r)]\!]})$$

Then, the hypothesis can be rewritten as:

$$(\sigma_1 : \theta_1 : s_1) \overset{\mathsf{GC}}{\mapsto} (\sigma_2 : \theta_2 : s_1) \overset{\mathsf{L}}{\mapsto} (\sigma_2 : \theta_2 : s_2)$$

If we take $(\sigma_4 : \theta_4 : s_3) = (\sigma_1 : \theta_1 : s_2)$, we can assert that:

$$(\sigma_1 : \theta_1 : \overset{\overset{s_1}{\equiv}}{E[\![r]\!]}) \overset{\mathsf{L}}{\mapsto} (\sigma_1 : \theta_1 : \overset{\overset{s_2}{\equiv}}{E[\![\sigma_2(r)]\!]})$$

because r is reachable in $(\sigma_1 : \theta_1 : s_1)$, and the $\overset{\mathsf{GC}}{\mapsto}$ step from the hypothesis preserves its binding, in the sense expressed in Lemma A.5: hence, if it was possible to perform the dereferencing in $(\sigma_2 : \theta_2 : s_1)$ (by hypothesis), it must be possible to perform it in $(\sigma_1 : \theta_1 : s_1)$, obtaining the same result. Finally, by preservation of bindings ready for collection after a $\overset{\mathsf{L}}{\mapsto}$ step, Lemma A.1, and the non-deterministic behaviour of $\overset{\mathsf{GC}}{\mapsto}$, we could ask for the GC step to remove exactly the necessary bindings so that $(\sigma_1 : \theta_1 : s_2) \overset{\mathsf{GC}}{\mapsto} (\sigma_2 : \theta_2 : s_2)$ holds. We obtained:

$$(\sigma_1 : \theta_1 : s_1) \overset{\mathsf{L}}{\mapsto} (\sigma_1 : \theta_1 : s_2) \overset{\mathsf{GC}}{\mapsto} (\sigma_2 : \theta_2 : s_2)$$

Finally, $(\sigma_3 : \theta_3 : s_2) \overset{\mathsf{rch}}{\sim} (\sigma_3' : \theta_3' : s_2)$ because

$$(\sigma_3 : \theta_3 : s_2) = (\sigma_2 : \theta_2 : s_2) = (\sigma_3' : \theta_3' : s_2)$$

- *The computation changes or dereferences locations from θ_1*: we would need to do case analysis on each computation that interacts with θ_1. As an example, let us consider table allocation. Then, the hypothesis can be rewritten as:

$$(\sigma_1 : \theta_1 : s_1) \overset{\mathsf{GC}}{\mapsto} (\sigma_2 : \theta_2 : s_1) \overset{\mathsf{L}}{\mapsto} (\sigma_2 : \theta_2 \uplus \{(tid, t)\} : \overset{\overset{s_2}{\equiv}}{E[\![tid]\!]})$$

for an adequate internal representation of a table, t, and table identifier tid, that, for our purposes, it will be useful if $tid \notin dom(\theta_1)$. If it is not the case, we can continue with our reasoning over an appropriate α-converted configuration, where the references in $(\sigma_1 : \theta_1 : s_1)$ are consistently changed so as to make $tid \notin dom(\theta_1)$. It is because of cases like this one that we cannot assert a stronger postponement statement, as the one in [?]: we are not talking about convergence towards a single configuration; we need to think in terms of $\overset{\mathsf{rch}}{\sim}$-equivalent configurations.
If we take

$$(\sigma_4 : \theta_4 : s_3) = (\sigma_1 : \theta_1 \uplus \{(tid, t)\} : s_2)$$

we know that:

$$(\sigma_1 : \theta_1 : s_1) \overset{\mathsf{L}}{\mapsto} (\sigma_1 : \theta_1 \uplus \{(tid, t)\} : s_2)$$

where we can ask for the instruction term to be exactly $s_2 = E[\![tid]\!]$. By Lemma A.1 we know that every binding which is ready for collection in $(\sigma_1 : \theta_1 : s_1)$ is in the same state in $(\sigma_1 : \theta_1 \uplus \{(tid, t)\} : s_2)$. Even more, such bindings just belongs to σ_1 or θ_1. Then, by the non-deterministic nature of $\overset{\mathsf{GC}}{\mapsto}$ we could ask for it to remove just the necessary bindings so as to make true

$$(\sigma_1 : \theta_1 \uplus \{(tid, t)\} : s_2) \overset{\mathsf{GC}}{\mapsto} (\sigma_2 : \theta_2 \uplus \{(tid, t)\} : s_2).$$

Then, the following holds:

$$(\sigma_1 : \theta_1 : s_1) \overset{\mathsf{L}}{\mapsto} \ldots \overset{\mathsf{GC}}{\mapsto} (\sigma_2 : \theta_2 \uplus \{(tid, t)\} : s_2)$$

Finally, $(\sigma_3 : \theta_3 : s_2) \overset{\mathsf{rch}}{\sim} (\sigma_3' : \theta_3' : s_2)$ because

$$(\sigma_3 : \theta_3 : s_2) = (\sigma_2 : \theta_2 \uplus \{(tid, t)\} : s_2) = (\sigma_3' : \theta_3' : s_2)$$

□

Correctness of simple GC. The expected statement of GC correctness should mention that, for a given configuration, the observations under $\overset{\mathsf{L}}{\mapsto}$ should be the same that those under $\overset{\mathsf{L+GC}}{\mapsto}$ (*i.e.*, $\overset{\mathsf{L}}{\mapsto} \cup \overset{\mathsf{GC}}{\mapsto}$). However, under $\overset{\mathsf{L}}{\mapsto}$ and $\overset{\mathsf{L+GC}}{\mapsto}$ we expect the observations to be just a singleton: the programs either diverge or reach to a end, returning some results or an error object. Giving this observation, we could change the statement of GC correctness to reach to a property that can be proved with less effort: given a configuration, under $\overset{\mathsf{L}}{\mapsto}$ its execution reaches to a end, if and only if its execution reaches to an end under $\overset{\mathsf{L+GC}}{\mapsto}$, and, in both cases, what is returned (either values or error objects) is the same.

The stated property will imply the preservation of observations, as defined in Definition 4.2, but it will allow us to focus just on convergent computations; preservation of divergent computations will be a consequence of the double implication structure of the statement:

Theorem A.11 (GC correctness). *For a given well-formed configuration $\sigma : \theta : s$,*

$$(\sigma : \theta : s) \Downarrow_{\overset{\mathsf{L}}{\mapsto}} (\sigma' : \theta' : s') \Leftrightarrow (\sigma : \theta : s) \Downarrow_{\overset{\mathsf{L+GC}}{\mapsto}} (\sigma'' : \theta'' : s'')$$

and $\mathsf{result}(\sigma' : \theta' : s') = \mathsf{result}(\sigma'' : \theta'' : s'')$.

Proof. Let us assume that $(\sigma : \theta : s) \Downarrow_{\overset{\mathsf{L}}{\mapsto}} (\sigma' : \theta' : s')$. Then $(\sigma' : \theta' : s')$ is a final configuration where result is defined. Because $\overset{\mathsf{L}}{\mapsto} \subseteq \overset{\mathsf{L+GC}}{\mapsto}$, it is always possible to emulate the previous trace by not using $\overset{\mathsf{GC}}{\mapsto}$ steps. Then, $(\sigma : \theta : s) \Downarrow_{\overset{\mathsf{L+GC}}{\mapsto}} (\sigma' : \theta' : s')$, where it follows that, in both cases, the computations returns the same, under $\overset{\mathsf{L+GC}}{\mapsto}$ and $\overset{\mathsf{L}}{\mapsto}$.

On the other hand, let us assume that

$$(\sigma : \theta : s) \Downarrow_{\overset{\mathsf{L+GC}}{\mapsto}} (\sigma' : \theta' : s')$$

Then, it must be the case that there exist a finite trace of computation steps, as follows:

$$(\sigma : \theta : s) \overset{\mathsf{L+GC}}{\mapsto} (\sigma_1 : \theta_1 : s_1) \overset{\mathsf{L+GC}}{\mapsto} \ldots \overset{\mathsf{L+GC}}{\mapsto} (\sigma_n : \theta_n : s_n)$$

where $(\sigma_n : \theta_n : s_n) = (\sigma' : \theta' : s')$ is a final configuration over which result is defined.

By applying inductive reasoning over the number of computation steps and the Postponement Lemma A.10, it can be shown that we can rewrite the previous trace as follows:

$$(\sigma : \theta : s) \overset{\mathsf{L}}{\mapsto} \ldots \overset{\mathsf{L}}{\mapsto} (\sigma_{i'} : \theta_{i'} : s_{i'}) \overset{\mathsf{GC}}{\mapsto} \ldots \overset{\mathsf{GC}}{\mapsto} (\sigma_{n'} : \theta_{n'} : s_{i'})$$

where every computation that does not involve GC is performed at the beginning. We obtained a convergent trace consisting only in $\overset{\mathsf{L}}{\mapsto}$ steps. That is:

$$(\sigma : \theta : s) \Downarrow_{\overset{\mathsf{L}}{\mapsto}} (\sigma_{i'} : \theta_{i'} : s_{i'})$$

What remains is to see if the result is also preserved. To that end, note that the postponement lemma used also tells us that

$$(\sigma_{n'} : \theta_{n'} : s_{i'}) \overset{\mathsf{rch}}{\sim} (\sigma_n : \theta_n : s_n)$$

Then, because final configurations which are $\overset{\mathsf{rch}}{\sim}$ represent the same result, according to Lemma A.4, it follows that

$$\mathsf{result}(\sigma_{n'} : \theta_{n'} : s_{i'}) = \mathsf{result}(\sigma_n : \theta_n : s_n)$$

Finally, because $\overset{\mathsf{rch}}{\sim}$ is closed under $\overset{\mathsf{GC}}{\mapsto}$ steps, Lemma A.7, it must be the case that:

$$(\sigma_{i'} : \theta_{i'} : s_{i'}) \overset{\mathsf{rch}}{\sim} (\sigma_{n'} : \theta_{n'} : s_{i'})$$

Hence,

$$\mathsf{result}(\sigma_{i'} : \theta_{i'} : s_{i'}) = \mathsf{result}(\sigma_{n'} : \theta_{n'} : s_{i'}) = \mathsf{result}(\sigma_n : \theta_n : s_n)$$

□

An immediate corollary of the previous theorem is that, under $\overset{\mathsf{L+GC}}{\mapsto}$, the set of observations over programs is a singleton, even under the non-determinism nature of $\overset{\mathsf{GC}}{\mapsto}$:

Corollary A.12. *For a well-formed configuration $\sigma : \theta : s$,* $|\mathsf{obs}(\sigma : \theta : s, \overset{\mathsf{L+GC}}{\mapsto})| = 1$

Proof. It follows immediately from the previous theorem and the determinism of programs under $\overset{\mathsf{L}}{\mapsto}$. □

Now, based on the observations of the beginning of this section, we can state an equivalent version of correctness for simple GC, but in terms of the notion of observations previously defined:

Corollary A.13 (GC correctness). *For a given well-formed configuration $\sigma : \theta : s$,*

$$(\sigma : \theta : s, \overset{\mathsf{L}}{\mapsto}) \equiv (\sigma : \theta : s, \overset{\mathsf{L+GC}}{\mapsto})$$

Proof. It follows directly from the previous corollary, together with Theorem A.11. Then, if $\mathsf{result}(\sigma', \theta', s') \in \mathsf{obs}(\sigma : \theta : s, \overset{\mathsf{L}}{\mapsto})$, for $(\sigma : \theta : s) \Downarrow_{\overset{\mathsf{L}}{\mapsto}} (\sigma' : \theta' : s')$, by theorem A.11, the previous occurs if and only if $(\sigma : \theta : s) \Downarrow_{\overset{\mathsf{L+GC}}{\mapsto}} (\sigma'' : \theta'' : s'')$, where

$$\mathsf{result}(\sigma', \theta', s') = \mathsf{result}(\sigma'', \theta'', s'')$$

Hence $\mathsf{result}(\sigma', \theta', s') \in \mathsf{obs}(\sigma : \theta : s, \overset{\mathsf{L+GC}}{\mapsto})$, and we can conclude that $\mathsf{obs}(\sigma : \theta : s, \overset{\mathsf{L}}{\mapsto}) = \mathsf{obs}(\sigma : \theta : s, \overset{\mathsf{L+GC}}{\mapsto})$. The converse is analogous.

If $\perp \in \mathsf{obs}(\sigma : \theta : s, \overset{\mathsf{L}}{\mapsto})$, by correctness of GC, it must happen if and only if $\perp \in \mathsf{obs}(\sigma : \theta : s, \overset{\mathsf{L+GC}}{\mapsto})$, and because of the determinism of both, $\overset{\mathsf{L+GC}}{\mapsto}$ and $\overset{\mathsf{L}}{\mapsto}$, we can conclude that:

$$\mathsf{obs}(\sigma : \theta : s, \overset{\mathsf{L}}{\mapsto}) = \mathsf{obs}(\sigma : \theta : s, \overset{\mathsf{L+GC}}{\mapsto})$$

The converse is analogous. □