Corrigendum to "Proving Highly-Concurrent Traversals Correct" by Feldman et al., Proceedings of the ACM on Programming Languages (PACMPL), Volume 4, Issue OOPSLA, Article No. 128

YOTAM M. Y. FELDMAN, Tel Aviv University, Israel
ARTEM KHYZHA, Tel Aviv University, Israel
CONSTANTIN ENEA, IRIF, Université de Paris, France
ADAM MORRISON, Tel Aviv University, Israel
ALEKSANDAR NANEVSKI, IMDEA Software Institute, Spain
NOAM RINETZKY, Tel Aviv University, Israel
SHARON SHOHAM, Tel Aviv University, Israel

In Proceedings of the ACM on Programming Languages (PACMPL), Volume 4, Issue OOPSLA, Article No. 128, pp 1–29 (Feldman et al., Proving Highly-Concurrent Traversals Correct), a mistake is present in code of the Logical-ordering tree example in Fig. 2. This was pointed out by Meyer et al. [2023].

A correct version of the code was deposed in a new version of the extended version [Feldman et al. 2020]. It concerns the order of updates in `insert`: the correct version is that `insert` first adds the new node to the list formed by following *succ* links, and only then to the list formed by *pred* links and the tree. The change affects lines 80–87 of Fig. 2. The correct `insert` code appears in Figure 1 below; the changed lines are marked in color.

The proof of the algorithm in Sections 3 and 6 already referred to the correct order, and is unchanged.

(See DOI: https://doi.org/10.1145/3428196)

REFERENCES

Dana Drachsler, Martin Vechev, and Eran Yahav. 2014. Practical Concurrent Binary Search Trees via Logical Ordering. In *PPoPP 2014*.

Yotam M. Y. Feldman, Artem Khyzha, Constantin Enea, Adam Morrison, Aleksandar Nanevski, Noam Rinetzky, and Sharon Shoham. 2020. Proving Highly-Concurrent Traversals Correct. *CoRR* (2020). https://arxiv.org/abs/2010.00911

Roland Meyer, Thomas Wies, and Sebastian Wolff. 2023. Embedding Hindsight Reasoning in Separation Logic. *Proc. ACM Program. Lang.* 7, PLDI (2023), 1848–1871. https://doi.org/10.1145/3591296

Authors' addresses: Yotam M. Y. Feldman, Tel Aviv University, Israel, yotam.feldman@gmail.com; Artem Khyzha, Tel Aviv University, Israel, artkhyzha@mail.tau.ac.il; Constantin Enea, IRIF, Université de Paris, France, cenea@irif.fr; Adam Morrison, Tel Aviv University, Israel, mad@cs.tau.ac.il; Aleksandar Nanevski, IMDEA Software Institute, Spain, aleks.nanevski@imdea.org; Noam Rinetzky, Tel Aviv University, Israel, maon@cs.tau.ac.il; Sharon Shoham, Tel Aviv University, Israel, sharon.shoham@gmail.com.

```
64 bool insert(int k)
65   x←tree-locate(k)
66   p←(x.key>k ? x.pred : x)
67   lock(p.succLock)
68   s←p.succ
69   if k∉(p.key,s.key] ∨ p.rem
70     restart
71   {{−∞} ⇝k p ∧ k ∈ (p.key, s.key]
       ∧ ¬p.rem ∧ p.succ = s }
72   if s.key=k
73     {{−∞} ⇝k s ∧ s.key = k ∧ ¬s.rem}
74     return false
75   n←new N(k)
76   n.succ←s
77   n.pred←p
78   z←chooseParent(p,s,n)
79   n.parent←z
80     {{−∞} ⇝k p ∧ ¬p.rem
        ∧ p.succ = s ∧ k ∈ (p.key, s.key)
        ∧ n.key = k ∧ ¬n.rem ∧ n.succ = s}
81   p.succ←n
82   s.pred←n
83   lock(z.treeLock)
84   if (z.key<k)
85     z.left←n
86   else
87     z.right←n
88   return true
```

Fig. 1. Corrected insert operation for a version of the Logical-ordering tree [Drachsler et al. 2014]. For brevity, **unlock** operations are omitted; a procedure releases all the locks it acquired when it terminates or **restart**s.