# ERRATA FOR "A STRATIFIED APPROACH TO LÖB INDUCTION"

DANIEL GRATZER AND LARS BIRKEDAL

## 1. The proof of the no-go theorem (Corollary 3)

While the no-go theorem (Corollary 3) is itself correct, the intermediate result used to justify it (Theorem 2) is not. We are grateful to Sean Moss for pointing out this oversight. We present a revised proof of Corollary 3 which parallels the techniques presented in Berger and Setzer [BS18].

We recall the statement of the no-go theorem (Corollary 3):

**Theorem 1.1.** *Conversion is undecidable in a guarded type theory satisfying the following requirements:*

- *there is a type $S$ equipped with a definitional isomorphism $S \cong \mathsf{nat} \times \blacktriangleright S$.*
- $\mathsf{next} : A \to \blacktriangleright A$ *is injective on closed terms,*
- *the* $\mathsf{lob}$ *operator unfolds.*

We fix some effective encoding of Turing machines $\mathtt{TM} = \mathsf{nat}$ as well as an encoding of the state of the tape of the machine as another natural number $\mathtt{State} = \mathsf{nat}$. We further assume the following operations are defined:

(1) $\mathtt{init} : \mathtt{TM} \to \mathtt{State}$
(2) $\mathtt{step} : \mathtt{TM} \times \mathtt{State} \to \mathtt{State}$
(3) $\mathtt{halt} : \mathtt{TM} \times \mathtt{State} \to \mathsf{bool}$
(4) $\mathtt{result} : \mathtt{State} \to \mathsf{nat}$

These operations parallel those required by Berger and Setzer [BS18] and are well-known to be definable in type theory—in fact, they are primitive recursive. Using these operations, we can define an element of $\mathtt{S}$ which represents the (potentially non-terminating) trace of a machine:

$$\mathtt{exec} : \mathtt{TM} \to \mathtt{S}$$
$$\mathtt{exec}(x) = \mathtt{go}(\mathtt{init}(x))$$
$$\quad \mathtt{go} : \mathtt{State} \to \mathtt{S}$$
$$\quad \mathtt{go} = \mathsf{lob}(r.\ \lambda s.\ \textbf{if}\ \mathtt{halt}(x,s)\ \textbf{then}\ \mathtt{const}(\mathtt{result}(s))\ \textbf{else}\ \mathtt{cons}(0, r \circledast \mathsf{next}(\mathtt{step}(x,s))))$$

Assuming that a machine $x$ terminates in $n$ steps with result $\epsilon$, and using the fact that $\mathsf{lob}$ unfolds, we therefore conclude the following:

$$\mathtt{exec}(x) = \mathtt{cons}(0, \ldots, \mathsf{next}(\mathtt{cons}(0, \mathsf{next}(\mathtt{const}(\epsilon)))))$$

Accordingly, using the $\eta$ law for dependent sums and the definitional isomorphism for $\mathtt{S}$ we see that if $x$ terminates with result 0 then $\mathtt{exec}(x) = \mathtt{const}(0)$. To see this, suppose $x$ terminates in 2 steps with result 0:

$$\mathtt{exec}(x)$$

*Date*: Wednesday 31st August, 2022.

$$\begin{aligned}
&= \mathsf{go}(s_0) && s_0 = \mathtt{init}(x) \\
&= \mathsf{cons}(0, \mathsf{next}(\mathsf{go}(s_1))) && s_1 = \mathtt{step}(x, s_0) \\
&= \mathsf{cons}(0, \mathsf{next}(\mathsf{cons}(0, \mathsf{next}(\mathtt{const}(0))))) \\
&= \mathsf{cons}(0, \mathsf{next}(\mathtt{const}(0))) && \text{Using the } \eta \text{ law and definition of } \mathtt{const} \\
&= \mathtt{const}(0)
\end{aligned}$$

This argument is easily seen to extend to programs taking an arbitrary but finite number of steps to terminate with result 0. Furthermore, if $x$ terminates in $n$ steps with result 1 then $\mathtt{exec}(x)$ is not definitionally equal to $\mathtt{const}(0)$; the former has $\mathsf{next}^n(1)$ for the $n$th element while the latter has $\mathsf{next}^n(0)$ and $\mathsf{next}$ is assumed to be injective on closed terms.

To complete our proof, we require the following classical result:[1]

**Theorem 1.2** (Rosser [Ros36], Kleene [Kle50], Trahtenbrot [Tra53])**.** *Consider the following subsets of natural numbers:*

$$A = \{x \mid x \textit{ is a Turing machine terminating with result } 0\}$$
$$B = \{x \mid x \textit{ is a Turing machine terminating with result } 1\}$$

*$A$ and $B$ are recusively inseparable. In other words, there is no computable function $\mathbb{N} \to \mathbb{N}$ which sends $A$ to $0$ and $B$ to $1$ while terminating on all inputs.*

We are now in a position to prove the no-go theorem:

*Proof of Theorem 1.1.* Suppose that conversion was decidable. Precisely, fix an effective encoding of terms of guarded type theory as natural numbers and assume there is a total computable function $e : \mathbb{N} \times \mathbb{N} \to \mathbb{B}$ which decides conversion.

In this case, consider the following computable function:

$$\lambda n.\ e(\ulcorner \mathtt{exec}(\mathsf{suc}^n(\mathsf{zero}))\urcorner, \ulcorner \mathtt{const}(0)\urcorner)$$

As $e$ is assumed to be total, this is a total computable function. However, we have just seen that this function separates Turing machines halting with 0 and those halting with 1, contradicting Theorem 1.2. □

## References


[BS18] Ulrich Berger and Anton Setzer. "Undecidability of Equality for Codata Types". In: *Coalgebraic Methods in Computer Science*. Ed. by Corina Cîrstea. Cham: Springer International Publishing, 2018, pp. 34–55. ISBN: 978-3-030-00389-0 (cit. on pp. 1, 2).

[Kle50] S. C. Kleene. "A symmetric form of Gödel's theorem". In: *Ind. Math* (1950), 12:244–246. DOI: 10.2307/2266709 (cit. on p. 2).

[Ros36] Barkley Rosser. "Extensions of Some Theorems of Gödel and Church". In: *The Journal of Symbolic Logic* 1.3 (1936), pp. 87–91. ISSN: 00224812. URL: http://www.jstor.org/stable/2269028 (visited on 08/08/2022) (cit. on p. 2).

[Tra53] B. A. Trahtenbrot. "On Recursive Separability". In: *Dokl. Acad. Nauk* 88 (1953), pp. 953–956 (cit. on p. 2).


[1] We have followed the attribution used by Berger and Setzer [BS18] for this result. There the authors also point to textbook reproductions of this fact.