Going nowhere and back: Is Trivialization the same as Zero Execution? [preprint]

Ivo Pezlar

Abstract

In this paper I will explore the question whether the Trivialization construction of transparent intensional logic (TIL) can be understood in terms of the Execution construction, specifically, in terms of its degenerate case known as the 0-Execution. My answer will be positive and the apparent contrast between intuitive understanding of Trivialization and 0-Execution will be explained as a matter of distinct yet related informal perspectives, not as a matter of technical or conceptual differences.

keywords: transparent intensional logic, execution, trivialization, construction, Tichý

1 Introduction and motivation

One of the most prominent features of transparent intensional logic (TIL, Tichý (1988), Duží et al. (2010), Raclavský et al. (2015)), and simultaneously the source of much puzzlement for TIL newcomers, is the Trivialization construction, which more or less acts as a procedural analogue of a constant. For example, an abstraction term $\lambda x.x + 1$ of lambda calculus with constants corresponds to a Closure construction $\lambda x[x\,'{+}\,'1]$ of TIL, where the prime symbol '$'$' indicates Trivialization constructions, namely Trivializations of the addition function $+$ and of the number 1, respectively.[1] A lot has been already written about Trivialization and its justification and I do not wish to add to that literature. Instead I want the explore the question whether Trivialization can be understood in terms of another TIL construction known as Execution, specifically, in terms of its degenerate case known as the 0-Execution (read as "Zero Execution"). Although it is not an issue of a great practical importance from the viewpoint of natural language semantics, it touches on fundamental theoretical aspects of TIL, which, I believe, make it a topic worthy of a closer look.

The goal of this paper, and I wish to emphasize this point, is not to argue for the removal of Trivialization from TIL, but to examine whether it can be

[1] In the literature, Trivialization is most commonly denoted by the superscript '^{0}' not by '$'$'. In this paper, however, I reserve the symbol '^{0}' for denoting 0-Execution. To ensure notational consistency I apply this convention to quotations as well.

understood as Execution, specifically, 0-Execution.[2] Ideally, it might also help TIL newcomers to more quickly grasp the main idea of the construction denoted by the unnecessary-looking but crucially important symbol '$'$'.

Finally, it is worth mentioning that the idea of treating Trivialization as 0-Execution is not new one and it has been floating around the TIL community for some time (at least since Raclavský (2003)) but only recently it gained more attention (Jespersen (2019), Pezlar (2019), Raclavský (2020)). In this paper, I want to reopen this issue, revisit some of the previous arguments, and offer a new perspective on the Trivialization vs. 0-Execution debate.

2 Trivialization vs. 0-Execution

Can we regard Trivialization construction in the terms of 0-Execution construction? Or more bluntly put, is Trivialization the same construction as 0-Execution or are they distinct? To get a better grasp of the issue, let us assume that Trivialization and 0-Execution are indeed two distinct constructions. Let us denote the instances of the former as $'X$ and of the latter as ${}^0X$ where X is any entity whatsoever (within the scope of the ramified type hierarchy of TIL).

Our task will be to show that every role that is typically played by Trivialization can also by played by 0-Execution. I will investigate this issue from two aspects: technical and conceptual. From the technical aspect, we want to check if every instance of $'X$ can be replaced by ${}^0X$ (without affecting the procedural behaviour of the construction in the sense that it still produces the same output in the same way, i.e., that it is still the same construction). From the conceptual aspect, we want to check if all the standard intuitions typically associated with Trivialization can be preserved when dealing with 0-Execution. In other words, do we lose something if we view Trivialization as 0-Execution?

If no fundamental discrepancies between Trivialization and 0-Execution of this sort are found, I believe we are warranted in claiming that Trivialization and 0-Execution can be understood as the same construction, just viewed from different perspectives, namely from the destination perspective and the path perspective, respectively (I will discuss these later).

For the purposes of this paper, from now on I will distinguish between Trivialization and Execution understood as different kinds of constructions, and Trivialization and Execution understood as specific instances of these kinds of constructions. I will write the former with capitalized first letters (Trivialization, Execution, ...) and the letter without the capitalization (trivialization, execution, ...). We can think of this distinction as similar to the distinction between axiom schemata and their particular instances. For example, we can say that $'5$, $'Alice$, and $'['5\ '+\ '7]$ are all trivialization constructions, or trivializations for short. Or, alternatively, we can say that $'5$, $'Alice$, and $'['5\ '+\ '7]$ are specific

[2] I use this opportunity to revisit one of my earlier discussions with prof. Marie Duží on the same topic. In the past, she was firmly against viewing Trivialization as 0-Execution and with this paper I would like to, if not to change her mind, at least to sow there a seed of doubt.

instances of the Trivialization construction kind (distinct from other construction kinds such as Closure, Composition, etc.). Thus, when Jespersen (2015) says that “Trivializations are the one-step or primitive or atomic constructions of TIL” (p. 329) from our viewpoint he talks about Trivialization as a general construction kind, but when Duží et al. (2010) say phrases like “the meaning of ‘cow’ (here the Trivialization $'Cow$)” (p. 231), from our viewpoint they talk about a specific instance of the Trivialization construction kind, namely $'Cow$. In short, in the first case we are talking about Trivialization understood as a kind of a construction, in the second case, we are talking about specific instances of thereof. This distinction might seem pedantic, but it will become useful later. Specifically, it will help us to better unpack some conceptual issues surrounding Trivialization and 0-Execution.

3 What is Trivialization?

Trivialization is an atomic construction, denoted as $'X$, and its main role is to pick definite objects that the compound constructions of TIL can operate on. Specific trivializations can be compared to constants, as they fulfil a similar role.

First, let us start by reviewing some of the standard specifications of Trivialization found in the literature. Tichý (1988) originally specified it as follows:

> Where X is any entity whatsoever, we can consider the trivial construction whose starting point, as well as outcome, is X itself. Let us call this rudimentary construction the *trivialization* of X and symbolize it as $'X$. To carry $'X$ out, one starts with X and leaves it, so to speak, as it is. (Tichý (1988), p. 63)

Compare with Duží et al. (2010):

> Trivializations match constants, by picking out definite entities in just one step. (Duží et al. (2010), p. 9)
>
> When X is an object of any type (including a construction), the Trivialization of X, denoted ‘$'X$’, constructs X without the mediation of any other constructions. $'X$ is the unique atomic construction of X that does not depend on valuation: it is a primitive, non-perspectival mode of presentation of X. (Duží et al. (2010), p. 43)
>
> It constructs X without any change. (Duží et al. (2010), p. 45)

And also with Raclavský (2020) (similarly in Raclavský et al. (2015)):

> Every construction is a mode of presentation of a certain object [...] $'X$ presents X as it is (without any change of X) (Raclavský (2020), p. 52)

Thus, we could say that the key aspect of Trivialization $'X$ is that it does nothing with X, that it does not change X in any way.[3]

[3] Later, I will argue that this naturally leads to viewing Trivialization as a degenerate case of Execution, specifically, 0-Execution.

Note. It is worth pointing out that Tichý (1988) considered even Trivializations and Executions of nonconstructions to have constituents ("Variables are the only *simple* constructions; all other constructions have constituent parts." Tichý (1988), p. 63). This is not the case in Duží et al. (2010) and later, where even Trivialization and Execution of nonconstructions are considered "partless", i.e., atomic (Duží et al. (2010)):

> **Definition 2.17 (*atomic construction*)** A *construction* C is *atomic* if C does not contain any other constituent but itself.
>
> *Corollary.* A construction C is atomic if C is
>
> - a variable; or
> - a Trivialization $'X$, where X is an entity of any type, even a construction; or
> - an Execution ${}^1X$ or a Double Execution ${}^2X$, where X is an entity of a type of order 1, i.e., a nonconstruction.
>
> An atomic construction of kind (i) or (ii) is v-proper for any valuation v. An atomic construction ${}^1X$, ${}^2X$, of kind (iii), is v-improper for any valuation v. In this case ${}^1X$ or ${}^2X$ does not v-construct anything, and ${}^1X \to \alpha$, ${}^2X \to \alpha$, for any type α, would constitute a type-theoretic mismatch. (Duží et al. (2010), pp. 247–248)

At first, it might seem somewhat surprising that we can encounter executions that are atomic constructions as well as executions that are nonatomic, i.e., compound constructions. I believe distinguishing between kinds of constructions and their specific instances (as discussed above) can help us here. Being atomic or compound construction is a property of executions understood as specific constructions, not of Executions understood as construction kinds. Thus, there is nothing extraordinary about the fact that we can have executions that are atomic, but also executions that are compound.

4 What is 0-Execution?

0-Execution is an atomic construction, denoted as ${}^0X$, and it is understood as the limiting case of Execution construction kind which also includes Single Execution[4] and Double Execution, denoted ${}^1X$, ${}^2X$, respectively. The main idea is that while Single Execution tells us to execute X once, and Double Execution tells us to execute X twice, or more precisely, to execute X and then execute again the result we obtain (if any), 0-Execution tells us not to execute X, just to leave it as it is. It is simply the degenerate case of Executions when the number

[4]Single Execution would be also an interesting topic for a further investigation as it seems often overlooked for its apparent "simplicity". In this paper, I will, however, not discuss it further.

of consecutive executions equals 0. Its main role is to tell us what should not be executed.[5]

In the literature, we can find the following specifications of 0-Execution. For example, Jespersen (2019) introduces it as follows:

> Since ${}^1X$ and ${}^2X$ are Single and Double Execution, respectively, it would be natural if ${}^0X$ was known as *Zero Execution*. In fact, whereas 'Trivialization' gives the wrong idea about Trivialization, which is anything but trivial, 'Zero Execution' sums up what ${}^0X$ is all about: ${}^0X$ *displays* X. If X is a procedure, then ${}^0X$ does not proceed to executing X. This is in fact the gist of Tichý's original definition that ${}^0X$ produces X without any change of X. In this paper, however, I will stick to the original term 'Trivialization' for continuity. (Jespersen (2019), p. 1320)

Similarly Pezlar (2019):

> If we should understand ${}^1X$ as 'execute X' (i.e., '^{1}' = one execution) and ${}^2X$ as 'execute X and then execute its result X' (i.e., '^{2}' = two executions), then, arguably the most natural reading of ${}^0X$—if we have never heard of trivialization—is 'do not execute X' (i.e., '^{0}' = zero executions). (Pezlar (2019), p. 203)

Thus, analogously to Trivialization, the key aspect of 0-Execution ${}^0X$ is that it does nothing with X.

5 Technical aspects

Assume that X is any entity whatsoever and that we have a trivialization of X, denoted $'X$, and a 0-execution of X, denoted ${}^0X$. We want to show that the procedural behaviour of $'X$ and ${}^0X$ is the same. There are eight cases in total we have to consider: first, whether X is a nonconstruction or a construction, second, if it is a construction, what kind of a construction. Although it might seem as unnecessary, I believe it will be instructive to present each case separately to better understand the relationship between 0-Execution and Trivialization.

1. X is a nonconstruction: $'X$ produces X, analogously ${}^0X$ produces X,
2. X is a construction:
   (a) X is a variable: $'x$ produces x, analogously ${}^0x$ produces x,
   (b) X is a closure: $'[\lambda x_1 \ldots x_m X]$ produces $[\lambda x_1 \ldots x_m X]$, analogously ${}^0[\lambda x_1 \ldots x_m X]$ produces $[\lambda x_1 \ldots x_m X]$,
   (c) X is a composition: $'[XX_1 \ldots X_m]$ produces $[XX_1 \ldots X_m]$, analogously ${}^0[XX_1 \ldots X_m]$ produces $[XX_1 \ldots X_m]$,

[5] Mind you, it does not mean that ${}^0X$ should not be executed, only that X itself should not be executed. However, using the standard conceptual framework of TIL, the X itself is "invisible", since 0-executions are considered to be atomic constructions (as are trivializations).

(d) X is a trivialization: $''X$ produces $'X$, analogously $^{0\prime}X$ produces $'X$,

(e) X is a 0-execution: $'^{0}X$ produces ^{0}X, analogously ^{00}X produces ^{0}X,

(f) X is a 1-execution (single execution): $'^{1}X$ produces ^{1}X; analogously ^{01}X produces ^{1}X,

(g) X is a 2-execution (double execution): $'^{2}X$ produces ^{2}X; analogously ^{02}X produces ^{2}X.

As we can see, both Trivialization and 0-Execution proceed in the same manner for all entities X: it just takes them and returns them without any change.

6 Conceptual aspects

So far it seems that the whole issue of Trivialization vs. 0-Execution is just a terminological dispute based on personal preference. Trivialization seems to be fully explainable, and thus replaceable, in terms of 0-Execution. This assessment would be, however, too rushed as it does not tell the whole story. Despite the considerations above, not all TIL researchers, prof. Marie Duží included, would agree that Trivialization can be viewed as 0-Execution.

What are the main objections against viewing Trivialization as 0-Execution? The five most important objections are, I believe, the following:

Objection 1: Trivialization binds free variables, Executions do not.

Reply: It is true that 1-Executions and higher do not bind free variables, but 0-Execution is not just another Execution. As we said, it is not only a limiting case but a degenerate case of Execution. And since degenerate cases exhibit qualitative differences to non-generate ones (e.g., a point can be considered as a degenerate case of a circle with radius 0), it should not be surprising that 0-Execution can have different properties than non-0-Executions. Especially, if these properties are necessary side effects of its specification. Recall that 0-Execution is essentially an instruction to "do nothing with X", to not change it in any way. If 0-Execution would not bind free variables, X might change with respect to a valuation or a substitution. And if so, it would no longer be 0-Execution, since something was done with X, it was not left as it was (see also Pezlar (2019), p. 205). Thus, the fact that 0-Execution binds free variables is not only unsurprising but to be expected.

Objection 2: Trivialization is always proper, Executions are not.

Reply: Analogous reasoning as above applies. In short, 0-Execution is a degenerate case of Executions and as such it exhibits different qualitative properties. The fact that 0-Execution is always proper construction can be explained with respect to its specification. The construction cannot fail, because, simply put, there is no possibility for it to do so: it just takes X and produces X back without any change.

Objection 3: Trivializations are always atomic constructions, but Executions can be atomic as well as compound.

Reply: True, but if anything, this fact rather supports the idea that Trivialization is a special case of Executions, since Executions appear to be a more general notion from this viewpoint: we can have atomic executions (including 0-executions) and compound executions, but we cannot have compound trivializations, only atomic ones.

Objection 4: Trivialization supplies entities for compound constructions, Executions do not.[6]

Reply: Analogous to replies to objections 1 and 2. In short, there is no reason why we cannot view the supplying of entities just as another degenerate aspect of 0-Execution.[7]

Objection 5: Trivialization is a dual operation[8] to Double Execution: Trivialization raises context, while Double Execution decreases it and Trivialization cancels out Double Execution.[9]

Reply: Statements like "Double Execution suppresses the effect of Trivialization. More generally, Double Execution decreases the level of a context." (Duží et al. (2010), p. 239) or "Unlike Trivialization, which is an operation of *mentioning*, Execution and Double Execution are operations of *using*." (Duží et al. (2010), p. 239) are not incorrect, but they are somewhat imprecise.

Why is that? Because it is not the case that Double Execution always suppresses the effect of Trivialization or that Trivialization is always an operation of mentioning. As a counterexample to the first statement, consider a construction $^{2\prime}X$ where X is a nonconstruction: here ^{2} cannot suppress $'$, because if it did, we would end up with just X which is a nonconstruction. In other words, the "cancelling-out" process would transform a construction into a nonconstruction, which certainly cannot be correct.[10] As a counterexample to the second statement, consider a construction $'X$ where X is a nonconstruction. And since

[6]See, e.g., "There are two atomic constructions that supply entities (of any type) on which complex constructions operate: *Variables* and *Trivializations*." (Duží et al. (2010), p. 42) This, however, also brings up the question what is the purpose of the other atomic (although improper) constructions such as ^1X when X is a nonconstruction.

[7]Furthermore, we should not forget that we can have compound constructions that contain no Trivialization, 0-Execution or Variable, e.g., a composition $[^1 5\ ^1{+}\ ^1 7]$. True, it is an improper construction, but still a construction. So, even though 1-Execution of nonconstructions and higher cannot supply entities on which compound constructions can operate, they can in a way indirectly refer to them (e.g., $^1 5$ effectively tells us that "5 is not a construction") and be used to construct compound, albeit improper, constructions.

[8]I use the term "operatation" in the sense of Duží et al. (2010), i.e., an operation understood as a process, not in its more standard sense as a function/mapping.

[9]See, e.g., Jespersen (2019), p. 1320.

[10]The fact that $^{2\prime}X$ is an improper construction does not change anything: improper constructions are still constructions.

mentioning is defined only for constructions (see Duží et al. (2010), p. 234), Trivialization cannot be here used for mentioning.

So, general statements like "Trivialization raises context" and similar are slightly misleading as they do not present the whole picture. The duality applies only to specific instances of Trivialization and Execution, namely to those of the form $'C$ and $^{2\prime}C$ where C is a construction. But these cases do not exhaust all possible instances of Trivialization and Execution. Hence, the raising/lowering of context is not an inherent property of Trivialisation/0-Execution construction kind, but only of their specific instances $'X/{}^0X$ where X is a construction. Thus, let us try to amend the original statements: trivializations of constructions raise context, while double executions decrease it and trivializations of constructions are cancelled by double executions.

Now, how can we explain this aspect of Trivialization, i.e., that it can *mention* constructions and thus give rise to hyperintensional contexts, in terms of 0-Execution? Again, the strategy is analogous as above: since 0-Execution is the degenerate case of Execution, special properties are to be expected. And, arguably, 0-Execution offers even more intuitive explanation of its ability to mention constructions than Trivialization does. Consider, e.g., a construction $\lambda w \lambda t[{}^0Calculates_{wt}\ {}^0Alice\ {}^0[{}^05\ {}^0+\ {}^07]]$ presented as a result of a semantic analysis of the sentence "Alice calculates 5 + 7". Since Alice is engaged in the calculation process itself and not in its result, the construction $[{}^05\ {}^0+\ {}^07]$ corresponding to this calculation should *not be executed* in the respective semantic analysis. And that is precisely what "$^0[{}^05\ {}^0+\ {}^07]$" stands for, i.e., it tells us to "execute $[{}^05\ {}^0+\ {}^07]$ zero times".

In short, it seems there is no property typically associated with Trivialization that could not be associated with 0-Execution and explained by the fact that 0-Execution is the degenerate case of Execution, hence it is bound to have different properties from 1-Execution and higher. Moreover, we have shown that these degenerate properties naturally arise from the specification of 0-Execution (e.g., binding of free variables as a necessary side effect of the fact that 0-execution ^0X should not change X in any way).

So what is Trivialization? Is it just an alternative title for 0-Execution? 0-Execution definitely seems to have enough distinctive properties (binding, properness, supplying objects, ...) that would warrant giving it a special name such as "Trivialization", similarly as, e.g., a set with a single element is also given a special name of "singleton". But if we choose to view Trivialization this way, we have to keep in mind that there is still 0-Execution running (or rather, Execution not running) under the hood of Trivialization, so to speak.

So far I have only discussed reasons why we can regard Trivialization as 0-Execution, however, I have not said much about why should we, i.e., what are the advantages of this approach. This is intentional, as my goal is not to argue for getting rid of the notion of Trivialization and relying instead solely on 0-Execution. I only wanted to show that Trivialization can be replaced by 0-Execution, that there is nothing that would prevent us from doing so, both from the technical and the conceptual viewpoints. That being said, there are,

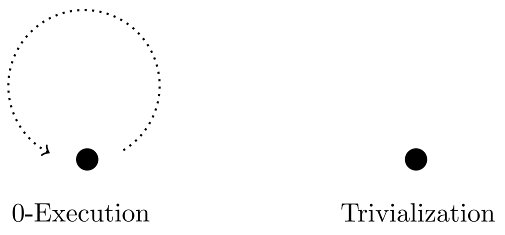

Figure 1: Difference between 0-Execution and Trivialization perspectives

I believe, some advantages of using 0-Execution, which I will briefly discuss in the concluding section.

7 Reconciliation

7.1 Two perspectives

In the previous section, we have seen that all the standard roles played by Trivialization can be also played by 0-Execution. Yet, declaring flatly that there is no difference between Trivialization and 0-Execution still does not seem entirely accurate, at least as far as the involved intuitions are concerned. I believe there is a difference between them, but it is a difference of informal perspectives that cannot be expressed within the conceptual framework of TIL.

To illustrate this difference, we will need to keep in mind two things: our treatment of 0-Execution as a degenerate case of Execution and Tichý's original informal explanation of Trivialization. Recall that according to Tichý, Trivialization is "the trivial construction whose starting point, as well as outcome, is X itself." (Tichý (1988), p. 63). So its construction "path" goes from X to X. But going from X to X can be also understood as not going at all, i.e., as starting from X and simply staying there. The following diagram (see fig. 1) should make this distinction clearer.

Thus, we could perhaps best understand 0-Execution as giving us the following informal instruction:[11] "start at A and go to B and $A = B$". On the other hand, Trivialization tells us: "start at A and do not go anywhere", or alternatively "stay at A". In both cases we end up at A, but the ways of how we got there are slightly different.

So, just as there is an intuitive difference between going out and returning to the same spot and not going out at all, so there is a difference between 0-Execution and Trivialization. It is not an important difference from a logical, or even a procedural point of view, but it can help us to explain the clash of intuitions associated with 0-Execution and Trivialization.

From this viewpoint, 0-Execution and Trivialization can be understood as

[11]In the style of Tichý's original informal explanation of Trivialization.

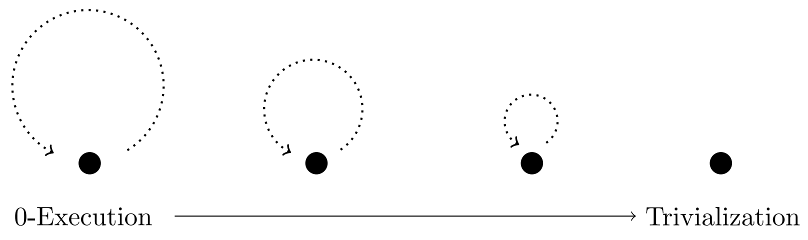

Figure 2: Transformation of the path perspective into the destination one

capturing two different perspectives of the corresponding underlying construction typically denoted as ${}^0X$. Furthermore, these two perspectives are not incompatible – we can easily imagine the "frivolous" path of 0-Execution as shrinking into the trivial path of Trivialization, which is just the starting point (similarly, as we can imagine a circle shrinking into a point). The following diagram (see fig. 2) depicts this process. Therefore, the difference between 0-Execution and Trivialization does not seem to be a simple matter of terminological preference but rather of a different conceptual perspectives, each emphasizing different aspects of the construction known as both 0-Execution and Trivialization. The former emphasizes its path, the latter its destination.

Are there some contexts where one perspective might be more suitable than the other? I believe so, e.g., when investigating foundational aspects of constructions it might be more beneficial to think of Trivialization in terms of 0-Execution. On the other hand, when dealing with more applied aspects of TIL, such as natural language analysis, opting for the Trivialization perspective might be more advisable, as in these cases we are not primarily interested in the inner workings of TIL constructions, they are mostly just a means to an end.

7.2 Pragmatic aspects

In this paper, I have focused on the technical and conceptual aspects but I have left out, arguably, the most decisive aspects determining the fate of 0-Execution – the pragmatic ones: habits, personal preferences, didactic considerations, and continuity. To these concerns I have little to say. If we prefer to use Trivialization instead of 0-Execution explicitly just for these pragmatic reasons, I have no issues with that. I just wanted to show that there is no technical nor conceptual reason that would justify this choice. Aside from the practical considerations, we seem to be free in choosing and switching between 0-Execution and Trivialization.

However, I would like to briefly discuss the last mentioned concern, the continuity. It is true that most of the TIL literature talks about "Trivialization", on the other hand, the notation itself does not change with the adoption of 0-Execution, it is just a matter of an informal retroactive interpretation whether we will read ${}^0X$ as 0-Execution or Trivialization. And most importantly, we do not need to get rid of Trivialization. We can keep it as it is – only with the added understanding that it is just a "disguised" Execution, specifically, its degenerate case of 0-Execution, and not a construction of its own kind. This newly acquired conceptual simplicity could lead to a system of TIL that is easier to learn (for example, TIL, as presented in Pezlar (2021), relies only on four basic constructions, namely, Variable, Closure, Composition, and n-Execution, instead of the standard six, i.e., Variable, Closure, Composition, Trivialization, Single Execution, and Double Execution). However, it might also have the opposite effect and produce a system that is more confusing, just as axiomatic systems with less axioms are not necessarily easier to understand and work with than systems with more axioms. Either way, this would be a matter for an empirical investigation.

8 Final remarks

Is Trivialization the same as 0-Execution? Based on the reasons given here, my answer is positive: yes, it is. There do not seem to be any convincing technical or conceptual reasons why Trivialization cannot be considered as a degenerate case of Execution. Of course, whether it is also a good idea to present it as such from a practical standpoint is another matter entirely. But if they are the same construction (i.e., given the same input, they produce the same output in the same way), how do we explain the fact that different intuitions seem to be associated with them? This, I believe, can be resolved by recognizing the two possible perspectives we can take in regards to 0-Execution/Trivialization, i.e., the path perspective and the destination perspective, with the former perspective being transformable into the latter: the "path" of 0-Execution can be contracted into a starting/ending point of Trivialization.

Acknowledgments. This work was supported by the Lumina quaeruntur fellowship awarded by the Czech Academy of Sciences (registration number: LQ300092101).

References

Marie Duží, Bjørn Jespersen, and Pavel Materna. *Procedural Semantics for Hyperintensional Logic: Foundations and Applications of Transparent Intensional Logic*. Springer, Dordrecht, 2010. doi: https://doi.org/10.1007/978-90-481-8812-3.

Bjørn Jespersen. Structured Lexical Concepts, Property Modifiers, and Transparent Intensional Logic. *Philosophical Studies*, 172(2):321–345, 2015. doi: 10.1007/S11098-014-0305-0.

Bjørn Jespersen. Anatomy of a Proposition. *Synthese*, 196(4):1285–1324, 2019. doi: https://doi.org/10.1007/s11229-017-1512-y.

Ivo Pezlar. On Two Notions of Computation in Transparent Intensional Logic. *Axiomathes*, 29(2), 2019. doi: 10.1007/s10516-018-9401-7.

Ivo Pezlar. Type Polymorphism, Natural Language Semantics, and TIL (ms.). 2021.

Jiří Raclavský. Executions vs. Constructions. *Logica et Methodologica*, 7:63–72, 2003.

Jiří Raclavský. *Belief Attitudes, Fine-Grained Hyperintensionality and Type-Theoretic Logic*. College Publications, London, 2020.

Jiří Raclavský, Petr Kuchyňka, and Ivo Pezlar. *Transparentní intenzionální logika jako characteristica universalis a calculus ratiocinator*. Masaryk University Press (Munipress), Brno, 2015.

Pavel Tichý. *The Foundations of Frege's Logic*. de Gruyter, Berlin, 1988.

Ivo Pezlar
Czech Academy of Sciences
Institute of Philosophy
Jilská 1, Prague 110 00
The Czech Republic
E-mail: `pezlar@flu.cas.cz`