Organon F 33 (3) 2026: 330–346 ISSN 2585-7150 (online)
https://doi.org/10.31577/orgf.2026.33305

RESEARCH ARTICLE

On Certain Problems of Collisionless Substitution into Double Execution in Transparent Procedural Logic (TPL)

Jozef Sábo*

Received: 21 June 2025 / Revised: 3 February 2026 / Accepted: 12 March 2026

Abstract: The article presents an analysis of rules for collisionless substitution in Transparent Procedural Logic, a new version of Transparent Intensional Logic. The article compares rules for collisionless substitution in Transparent Procedural Logic and the current standard form of Transparent Intensional Logic. The article shows that, in some cases, the current rule for collisionless substitution into double execution in Transparent Procedural Logic creates problems in determining the object v-constructed by the resulting construction.

Keywords: Collisionless substitution; construction; free occurrence of variable; Transparent Intensional Logic; Transparent Procedural Logic; the ramified hierarchy of types.

1. Introduction

Transparent Intensional Logic (*TIL*) is a logical system developed by Tichý (1988). Later, Materna, Duží and Jespersen (2010) developed it into a standard variant of TIL 2010 *(TIL 2010)*.

* Institute of Philosophy SAS, v.v.i
https://orcid.org/0000-0001-8182-2679
Department of Analytic Philosophy, Institute of Philosophy SAS, v.v.i., 3 Klemensova 19, Bratislava, 811 09, Slovak republic
jozef.sabo@savba.sk

Recent research on TIL has explored hyperintensional contexts (Duží 2012) and substitution into hyperintensional contexts (Duží et. al. 2010; Duží and Jespersen 2012). TIL 2010 was used to analyse offices (Duží et. al. 2011; Duží and Číhalová 2024), empty names (Kosterec 2023), as a system to investigate structural isomorphism of meaning (Duží 2014), and more.

Kosterec (2020; 2021; 2024, chapter 2.4.3) proved several inconsistencies in TIL 2010, which pose a challenge to the results formulated within that system.[1]

These problems were addressed in the subsequent phase of the development of TIL through two approaches. The first was the development of an updated version of TIL 2010 by Duží and Jespersen (Duží and Jespersen 2025), labelled as TIL 2025 (*TIL 2025*)[2]. The second approach to the problems identified in TIL 2010 is Kosterec's system of TIL, introduced under the name Transparent Procedural Logic (*TPL*) (Kosterec, 2024).

TPL shares its basic definitions with Tichý's original formulation of TIL. However, it introduces substantial modifications to the definition of collisionless substitution and to the related notions of displayed/executed and v-executed/v-displayed occurrence of construction. As a result, TPL differs substantially from TIL 2010 and TIL 2025. This paper is concerned exclusively with TPL and does not provide an analysis of TIL 2025.

Collisionless substitution is the main logical operation on which many core theorems of transparent logics depend[3]. In this article, I demonstrate that TPL does not have clear rules describing changes in underlying constructions in some instances of collisionless substitution into double

[1] The main alternative to TIL 2010 has been Transparent Hyperintensional Logic *(THL)* (Raclavský et. al. 2015, Raclavský 2020). Kosterec (2024, chapter 2.3.5.) proved that similar inconsistencies as in TIL 2010 occur also in THL.

[2] TIL 2025 differs from TIL 2010 primarily with respect to the definition of Closure and by adopting a more restrictive definition of free variables. In all other respects, TIL 2025 retains the core definitions of TIL 2010 as well as its underlying computational mechanism.

[3] These core theorems are *Theorem: Replacement of free Variables, Theorem: Compensation Principle, Theorem: Restricted validity of β-reduction by name, Theorem: Validity of β-conversion by value*, and *Theorem: Validity of α-conversion* (Duží et. al. 2010; Kosterec 2024, chapter 2.2.3.)

execution. I also show that current rules for collisionless substitution into double execution in TPL fails to align with rules governing v-constructing for a double execution in TPL.

This article is structured as follows. In the first part, I briefly introduce the concept of construction with a focus on double execution. The next section explains the concept of collisionless substitution in TPL and how it differs from TIL 2025. Next, I present an example illustrating specific problems in applying the rule for collisionless substitution into double execution in TPL. The article concludes with a brief summary of the main findings.

2. Double Execution in TPL/TIL 2025

The basic concept of TIL/TPL is that of *construction;* in TIL 2025, this concept is denoted under the term procedure.[4] A construction is an abstract procedural object that represents how to obtain a result by a particular procedure [TIL 2025 defines six kinds of procedures: *Trivialization, variable, Execution, Double Execution, Closure, and Composition*] [5] Constructions produce an object through v-constructing.[6] However, it does not mean that every construction must v-construct something every time.[7] Some constructions do not v-construct anything—they are like dead ends. This feature

[4] The underlying definitions of procedures in TIL 2025 are largely identical to the definitions of construction in TIL 2010, with the exception of certain modifications concerning *Closure* (Duží and Jespersen 2025, 21).

[5] In this article, I do not provide the complete definitions of constructions/procedures and v-constructing/v-producing in TIL/2010/TIL 2025. These definitions have already been presented and discussed in several published works, including (Duží and Jespersen 2012; Duží et. al. 2010; Jespersen and Duží 2022; Kosterec 2024; Duží and Jespersen 2025). Therefore, I limit the presentation of the various technical aspects of TPL/TIL 2025 only to the extent necessary for the topic discussed in this article.

[6] This process of semantic computation of the value of a procedure is called v-producing in TIL 2025 (Duží and Jespersen 2025, 21).

[7] A letter *v* in the term v-constructing means that constructing is performed under a particular valuation. Valuation is a technical notion in TPL/TIL 2025, a total

enables modelling partial functions in TPL/TIL 2025.[8] A construction that does not v-construct anything for a given set of parameters [referred to as a *valuation* in TPL/TIL 2025] is called v-improper. If a construction never v-constructs an object under any possible valuation, it is referred to as an improper full-stop (Kosterec, 2024, 29).

TPL adopts the same definitions of construction and v-constructing as Tichý's TIL (Kosterec, 2024, chapter 2.4). These definitions match those in TIL 2025, except for the τ-closure [instead of this type of construction, TIL 2025 defines a similar but different type of procedure called *Closure*]. However, this difference between Closure in TIL 2025[9] and τ-closure in TPL is not relevant to the analysis undertaken in the present article.

Double execution is a construction/procedure in TPL/TIL 2025 of the form ^2X, where X is any entity.

Example 1. Let us have a variable x [also a construction/procedure in TPL/TIL 2025]; then ^2x is a double execution. The double execution ^2x is a distinct construction from the variable x.

Double execution v-constructs a particular object (if any) according to the following rule:

Definition 1. TPL: Def. v-constructing
Let v be a valuation. Then:
iv) If X is a construction and X v-constructs a construction Y, then ^2X v-constructs the object (if any) that is v-constructed by Y. Otherwise, ^2X does not v-construct anything.

This definition implies that, for the double execution ^2X to v-construct anything, these conditions must be met at the same time: i) the entity X must be a construction, ii) the construction X must v-construct an object Y that is also a construction, and iii) the construction Y must v-construct

function that attributes value to all variables (Kosterec, 2024, 30; Duží and Jespersen 2025, 11).

[8] TIL 2025 and TPL are defined over the set(s) of partial functions.

[9] TIL 2025 employs a modified version of Closure, which differs from Closure in TIL 2010. This difference has significant impact on semantic computation with Closure in TIL 2025 (Duží and Jespersen 2025, 21).

an object. If these conditions are met, then the double execution ^2X v-constructs the same object as the one v-constructed by the construction Y. Otherwise, the double execution^2X does not v-construct anything.

Example 2. double executions ^2house and $^2\mathbf{1}$ do not v-construct anything under any valuation. Why? Because object $house$ (physical object from bricks and mortar) and number $\mathbf{1}$ (mathematical number) are not constructions. Therefore, these double executions are improper full-stop.

What does the double execution ^2x v-construct? From definition *(iv) TPL: Def. v-constructing* follows that an object v-constructed by the double execution ^2X depends on the object that is v-constructed by the construction Y, which is a construction v-constructed by the construction X. Therefore, the answer to this question depends on the object attributed by a valuation v to the variable x. If, according to a valuation v, the variable x does v-construct a construction that is v-improper or improper full stop, then the double execution ^2x is v-improper.

Example 3.[10] Let us have valuation v, according to which the variable x v-constructs the double execution $^2\mathbf{1}$. Then, under the valuation v, the double execution ^2x is v-improper.

Remark 1. The notation $v(^2\mathbf{1}/x)$ indicates that the variable x, under the valuation v, has a value of $^2\mathbf{1}$, i.e.,the variable x v-constructs the double execution $^2\mathbf{1}$

Example 4.[11] Let us have another valuation v, according to which the variable x v-constructs a variable y and this variable y v-constructs number $\mathbf{1}$. Then, according to this valuation v, the double execution ^2x v-constructs the number $\mathbf{1}$.

[10] In Example 3, the double execution ^{2}1 is the construction of order 1 and higher over B, the variable x is the construction of order 2 and higher over B and the double execution ^{2}x is the construction of order 3 and higher over B in the ramified hierarchy of types.

[11] In Example 4, the variable y is the construction of order 1 and higher over B, the variable x is the construction of order 2 and higher over B and the double execution ^{2}x is the construction of order 3 and higher over B in the ramified hierarchy of types.

Remark 2. A sign $\rightarrow_v$ stands for v-constructing/v-producing in TPL/TIL 2025; therefore, we can represent Example 4 as follows: $x \rightarrow_v y$, $y \rightarrow_v 1$, ${}^2x \rightarrow_v 1$.

The construction Y [in Example 4, the variable y] [12] may not be explicitly expressed in the construction ${}^2X$ [in Example 4, the double execution ${}^2x$]. If the construction Y is not explicitly expressed in construction ${}^2X$, then we may say that the double execution ${}^2X$ is "built upon" the hidden layer formed by construction Y. Moreover, a double execution can even be 'built' on several layers of other constructions. This occurs when the construction X in the double execution ${}^2X$ v-constructs a construction that is itself double execution, i.e. $X \rightarrow_v {}^2Y$. If this is the case, then to determine an object v-constructed by the double execution ${}^2X$, it is necessary to consider additional construction v-constructed by the construction Y.

Example 5.[13] Assume valuation v: $x \rightarrow_v {}^2y$, $y \rightarrow_v z$ and $z \rightarrow_v 1$. From $y \rightarrow_v z$ and $z \rightarrow_v 1$ follows that ${}^2y \rightarrow_v 1$ [according to (iv) TPL: Def. v-constructing an object v-constructed by the double execution ${}^2y$ is the same as v-constructed by a variable z]. Therefore, ${}^2x \rightarrow_v 1$ [from (iv) TPL: Def. v-constructing follows that an object v-constructed by ${}^2x$ is the same as an object (if any) v-constructed by ${}^2y$].

To determine the identity of an object v-constructed by the double execution ${}^2x$, only the object v-constructed by the double execution ${}^2y$ is considered. The definition *(iv) TPL: Def. v-constructing* does not establish

[12] Construction Y is expressed in ²X if X is a *Trivialization*. "TIL being a hyperintensional system, each construction C can occur not only in Execution mode to produce an object (if any) when being executed but also as an object in its own right on which other (higher-order) constructions operate. The Trivialisation of C causes C to occur just presented as an argument, as mentioned above. Yet sometimes, we need to cancel the effect of Trivialisation and trade the mode of C for Execution mode." (Duží and Číhalová 2024)

[13] In Example 5, the variable z is the construction of order 1 and higher over B, the variable y is the construction of order 2 and higher over B, the double execution ²y is the construction of order 3 and higher over B, the variable x is the construction of order 4 and higher over B and the double execution ²x is the construction of order 5 and higher over B in the ramified hierarchy of types.

a direct connection between the value of the double execution ^2x and the value of the variable z. The identity of an object v-constructed by the double execution ^2x depends on the value of the variable z only because an object v-constructed by the variable z determines an object v-constructed by the double execution ^2y.

If, under some alternative valuation v_i, the variable y would not v-construct the variable z but instead a variable z_i; then this change would affect ^2x only because an object v-constructed by the double execution ^2y would depend on an object v-constructed by the variable z_i. Then, according to this new valuation v_i, an object v-constructed by the variable z_i and not the variable z would be relevant to the identity of an object v-constructed by the double execution ^2y and therefore the double execution ^2x.

What if the variable y does not v-construct the variable z, but a double execution ^2z instead? In this case, an additional "layer" of constructions is needed to determine an object v-constructed by the double execution ^2x.

Example 6. [14] Let us modify Example 5: $y \rightarrow_v {}^2z$, $z \rightarrow_v r$, $r \rightarrow_v 1$. If this is the case, then $^2y \rightarrow_v 1$ [from (iv) TPL: Def. v-constructing follows that the double execution ^2z v-constructs the same object as v-constructed by the variable r, and the double execution ^2y v-constructs the same object as the double execution ^2z, namely, the number 1]. At the same time, if $x \rightarrow_v {}^2y$, then $^2x \rightarrow_v 1$ [from (iv) TPL: Def. v-constructing follows that an object v-constructed by the double execution ^2x is the same as v-constructed by the double execution ^2y, namely, 1].

The layered structure of double execution may create difficulties in the application of the rules for collisionless substitution in TPL. These difficulties are illustrated by an example in the following section of the article.

[14] In Example 6, the variable r is the construction of order 1 and higher over B, the variable z is the construction of order 2 and higher over B, the double execution ^{2}z is the construction of order 3 and higher over B, the variable y is the construction of order 4 and higher over B, the double execution ^{2}y is the construction of order 5 and higher over B, the variable x is the construction of order 6 and higher over B and the double execution ^{2}x is the construction of order 7 and higher over B in the ramified hierarchy of types.

3. Collisionless Substitution into Double Execution in TPL

Collisionless substitution in TPL has new rules that differ significantly from those in TIL 2010/TIL 2025. The full definition of collisionless substitution in TPL is as follows:

Definition 2. TPL: Def. collisionless substitution

Let x be a variable, and C and D are any kinds of construction. If x is not free in C, then *the result of substituting* D *for* x *in* C is C. Assume that x is free in C. Then:

(a) If C is x, then *the result of substituting* D for x in C is D. If C is ${}^1X$ or ${}^2X$ then *the result of substituting* D for x in C is ${}^1Y$, ${}^2Y\{x := D\}$, *where* Y *is the result of substituting* D *for* x *in* X. ${}^2Y\{x := D\}$ *means that all of the free occurrences of* x *in* ${}^2Y$ *are replaced with* D.

(b) If C is $[XX_1 \ldots X_m]$, then *the result of substituting* D for x in C is $[YY_1 \ldots Y_m]$, where $Y, Y_1, \ldots, Y_m$ are *the results of substituting* D for x in X, $X_1, \ldots, X_m$, respectively.

(c) Let C be of the form $[\lambda_T x_1 x_m Y]$; for $1 \leq i \leq m$, let $y_i = x_i$ if x_i is not free in D, and otherwise let $y_i =$ the first variable that v-constructs entities of the same type as x_i, does not occur or $\boldsymbol{v}$*-occur* in C, is not free in D, and is distinct from $y_1, \ldots, y_{i-1}$. Then, *the result of substituting* D for x in C is $[\lambda y_1 \ldots y_m Z]$, where Z is the result of substituting D for x in the result of substituting y_i for $x_i (1 \leq i \leq m)$ in Y.

I now provide several examples to better illustrate how collisionless substitution operates in TPL/TIL 2025. Then, I analyse the rule *(a) TPL: Def. collisionless substitution* for collisionless substitution into double execution in detail.

The definition of a free variable is essential to the functioning of collisionless substitution. The first line of *TPL: Def. collisionless substitution* addresses cases in which the variable x, for which the construction D is substituted, is not free in C. If this is the case, then collisionless substitution does not affect the construction C in any way. The result of collisionless substitution of the construction D for the variable x is identical to the construction C.

Remark 3. The notation $C(D/x)$ indicates the collisionless substitution of the construction D for the variable x in the construction C.

Example 7. Let us have valuation v, the construction ${}^0x$, and the construction D of any type. The construction ${}^0x$ is called trivialization in TPL/TIL 2025. The variable x is not free in trivialization ${}^0x$ [the variable x is *triv*-bound or bound by trivialization in trivialization ${}^0x$]. Therefore, the collisionless substitution ${}^0x(D/x)$ results in the construction ${}^0x$ [by the first line of TPL: Def. collisionless substitution [2]].

Before proceeding, we provide one further example of collisionless substitution in which the variable x is free in the construction C.

Example 8. Let us have valuation v, the variable x and the construction D of any type. Each variable is free in itself. Therefore, the variable x is free in the construction x. A collisionless substitution of the construction D for the variable x results in the construction D [this result follows from the first line of the rule (a) *TPL: Def. collisionless substitution*].

I will now test the application of the rule for collisionless substitution into double execution in TPL by the following example:

Example 9.[15] Assume valuation v, under which:

- $2/\tau$
- $x/{}^*1 \rightarrow_v 2$
- $q/{}^*2 \rightarrow_v x$
- ${}^2q/{}^*3 \rightarrow_v 2$

Remark 4. A sign / assigns a type from the ramified hierarchy of types to an entity; in Example 9, these types are: $\tau, {}^*1, {}^*2, {}^*3$.[16]

[15] In Example 9, the variable x is the construction of order 1 and higher over B, the variable q is the construction of order 2 and higher over B and the double execution ^{2}q is the construction of order 3 and higher over B in the ramified hierarchy of types.

[16] TIL is an open-ended system. Type τ designates a set of real numbers in the objectual base B. Types *1,*2,*3 designates types of constructions in the ramified hierarchy of types. I do not discuss these types in more detail. There are several

In Example 9, the variable q v-constructs the variable x. Therefore, an object v-constructed by the double execution ^2q is identical to an object v-constructed by the variable x, namely, the number 2. The variable x is not explicitly expressed in the body of the variable q, or the double execution ^2q. In this sense, the variable x is "hidden" within the underlying structure of the double execution ^2q.

The collisionless substitution in TIL 2025 affects only explicit parts (sub-procedures) of procedures. Therefore, the result of the collisionless substitution $^2q(D/x)$ is the procedure ^2q itself in TIL 2025 [since the variable x is not a sub-procedure of the double execution ^2q, it does not have a free occurrence in ^2q (Duží and Jespersen 2025, 22).[17]

At TPL, the situation differs significantly. TPL introduces a new definition of free variable and free occurrence of variable that differs from the definitions adopted by TIL 2010/TIL 2025. According to this new definition, the variable x has free occurrence in the double execution ^2q in TPL.

Furthermore, rule *(a) TPL: Def. collisionless substitution* allows substituting for the free occurrence of a variable x, not only in the double execution ^2X, but also in constructions forming "hidden" layers of double execution ^2X. Therefore, even if a variable x is not a subconstruction of double execution ^2X, we can still substitute for free occurrences of this variable x in constructions underlying the double execution ^2X.

> *Nowhere in the definition is it stipulated that x must be a subconstruction of C. Therefore, the definition also covers v-executed/v-displayed occurrences as soon as we consider them free/bound.* (Kosterec 2024, 120).

sources that provide a detailed explanation of the ramified hierarchy of types (Duží et. al. 2010; Jespersen and Duží 2022, Kosterec 2024; Duží and Jespersen 2025).

[17] This follows from rule *vi) of Definition 4.7 (Correct collision-less substitution)*, which states that: *Let x be a variable and C, D procedures. If x is not free in C, then the result of substituting D for x in C is C.* Hence, let the occurrence of x be free in C. vi) If C is ^{2}X where X is a procedure, then the result of the substitution of D for x in C is ^{2}Y, where Y is the result of the substitution of D for x in X. (Duží and Jespersen 2025, 24).

This change in the definition of collisionless substitution together with the new definition of free variable [18] constitutes a distinction between TPL and TIL 2025. [19]

By rule *(a) TPL: Def. collisionless substitution*, the result of collisionless substitution ${}^2q(D/x)$ is ${}^2q\{x := D\}$, which means that all free occurrences of the variable x in the double execution ${}^2q$ are replaced by the construction D.

The definition of collisionless substitution in TPL clearly states that in the construction ${}^2q\{x := D\}$,[20] the variable x is to be replaced by the construction D, but what exactly does it mean? In Example 9, the variable x is not a subconstruction of the double execution ${}^2q$ or the variable q. Therefore, the construction D cannot replace free occurrences of the variable x in those constructions. What are other alternatives? Kosterec is not clear on this point, and *(a) TPL: Def. collisionless substitution* is open to several interpretations in that regard.

To further elaborate on this point, instead of the construction D, we consider a collisionless substitution of construction ${}^{12}q$ in Example 9. This

[18] All these changes in the TPL definitions concern the so-called secondary definitions. Kosterec distinguishes between primary and secondary definitions in transparent logics. Primary definitions concern key notions of any system of transparent logic. Primary definitions are definitions of the following: *valuations, constructions, constructing according to valuation, and the ramified hierarchy of types*. Primary definitions are essential for characterizing a logical system as a TIL. According to Kosterec, secondary definitions must correspond to the core definitions of the system. Secondary definitions are *Def. Displayed vs. executed occurrence of a construction, Def. (v-displayed vs. v-executed occurrence of a construction), Def. free variable and Def. collisionless substitution*. The main differences between various transparent logics lie in formulating secondary definitions (Kosterec 2024, 80)

[19] In the following examples, I only state whether a certain variable is free or not according to the rules of TPL. A detailed discussion on the definition of free variable in TPL is not necessary for the topic discussed in the article.

[20] The TPL introduces new notation {}, which is not present in TIL. *"It is useful to have a notation to distinguish between constructions that are considered the result of a substitution for variables and those that are not – to make the substitution explicit. I suggest that we use a notation similar to the standard notation for substitution from lambda calculi."* (Kosterec 2024, 173)

kind of construction is called single execution in TPL/TIL 2025. [21] The single execution ^{12}q v-constructs the same object (if any) that is v-constructed by the double execution ^{2}q, that is, the number 2 [this follows from *(iii) TPL: Def. v-constructing*].[22]

The single execution ^{12}q is chosen deliberately because it contains the double execution ^{2}q as its subconstruction. Therefore, the collisionless substitution $^{2}q(^{12}q/x)$ includes the substitution of the double execution ^{2}q into itself. My challenge with this example is to examine whether the rule *(a) TPL: Def. collisionless substitution* can prevent a vicious cycle in this case.

A vicious cycle occurs when a construction v-constructs itself [or another construction that determines an object v-constructed by this construction]. An example of a vicious cycle is a situation in which $x \rightarrow_v x$. Transparent logics are typed lambda calculi and prohibit these types of constructions by employing a type system called *the ramified hierarchy of types.*[23] Any construction in which a vicious cycle of this kind occurs cannot be correctly typed and thus stands outside the ramified hierarchy of types.

What is the result of the collisionless substitution of the single execution ^{12}q for the variable x in the double execution ^{2}q in TPL? TPL threats the variable x as free in the double execution ^{2}q. Therefore, according to rule *(a) TPL: Def. collisionless substitution,* the result of the collisionless substitution $^{2}q(^{12}q/x)$ is a construction $^{2}q\{x := {}^{12}q\}$, which means that all free occurrences of x in ^{2}q are replaced with ^{12}q.

The fact that the variable x is not expressed in the double execution ^{2}q or the variable q means that the single execution ^{12}q cannot replace the variable x in either the double execution ^{2}q or the variable q. Yet, according to *(a) TPL: Def. collisionless substitution*, the collisionless substitution of ^{12}q for the variable x in the double execution ^{2}q must affect the resulting

[21] According to the valuation v in Example 9, the single execution ^{12}q/*4 is the construction of order 4 and higher over B.

[22] This rule states that "*If* X *is a construction, then* ^{1}X v-*constructs the object (if any) that is* v-*constructed by* X. *Otherwise,* ^{1}X *does not* v-*construct anything"* (Kosterec 2024, 45; Kosterec 2024, 168).

[23] TPL and TIL 2025 employ nearly identical rules for the determination of type in the ramified hierarchy of types (Kosterec 2024, 168; Jespersen and Duží 2025, 13).

construction ${}^{2}q\{x := {}^{12}q\}$ in some meaningful way. Therefore, the collisionless substitution ${}^{2}q({}^{12}q/x)$ should yield some changes that would differentiate the construction ${}^{2}q\{x := {}^{12}q\}$ from the double execution ${}^{2}q$.[24] Kosterec does not explain the application of this rule in detail, and to my best knowledge, no material published on TPL discusses this particular example.

There are two possible approaches to the explanation of the rule *(a) TPL: Def. collisionless substitution*. The first possibility is to assume that collisionless substitution for variables that are "hidden" in deeper layers of double execution directly changes the structure of affected constructions.

If we accept this explanation, the single execution ${}^{12}q$ replaces the variable x as an object v-constructed by the variable q in the construction ${}^{2}q\{x := {}^{12}q\}$. Then, we would need to determine an object v-constructed by the construction ${}^{2}q\{x := {}^{12}q\}$ under a modified valuation according to which $q \rightarrow_v {}^{12}q$ [instead of $q \rightarrow_v x$ that was stipulated by valuation v in Example 9]. I refer to this explanation as *a static explanation*. If this were the case, then from $q \rightarrow_v {}^{12}q$ it would follow that the object v-constructed by the construction ${}^{2}q\{x := {}^{12}q\}$ is identical to the object v-constructed by the single execution ${}^{12}q$ [according to rule *(iv) TPL: Def. v-constructing*]. The advantage of *the static explanation* is that it conforms to the general rule for v-constructing for double execution in TPL *(iv) TPL: Def. v-constructing*. In most cases, this explanation would seem preferable, apart from this example, when it inevitably leads to a vicious cycle due to $q \rightarrow_v {}^{12}q$. The variable q cannot be properly typed according to the condition $v({}^{12}q/q)$; therefore, it must stand outside the ramified hierarchy of types. Consequently, under *the static explanation*, the resulting construction ${}^{2}q\{x := {}^{12}q\}$ would stand outside the ramified hierarchy of types as well [because $q \rightarrow_v {}^{12}q$ is not a construction within the ramified hierarchy of types, it does not v-construct any object; therefore, the construction ${}^{2}q\{x := {}^{12}q\}$ would not v-construct any object either]. This result is not welcome because it violates the compensation principle in TPL.[25] From this it follows that the

[24] Kosterec (2024, 173) explicitly reaffirms this observation: *"The result of the collisionless substitution of* D *for* x *in* C *is to be followed by* {x := D}*. Then* ^{20}x *is not equivalent to* ^{20}x{x := D}*."*

[25] *Compensation principle*: Let C be a construction. Then for any valuation v and a construction D, if D v-constructs an entity d, then C(D/x) v-constructs an entity

construction D cannot be the object v-constructed by X or any other construction, in which D replaces free occurrences of x, in the construction of the form ${}^2Y\{x := D\}$. Therefore, *the static explanation* must be rejected.

The second possible explanation of rule *(a) TPL: Def. collisionless substitution* is to assume that collisionless substitution affects the identity of an object v-constructed by a double execution during individual steps of v-constructing. I refer to this explanation as *a dynamic explanation*. In *the dynamic explanation*, it is still true that $q \rightarrow_v x$ in ${}^2q\{x := {}^{12}q\}$. Only after this step of v-constructing happens [when the variable x has been v-constructed by the variable q], the single execution ${}^{12}q$ instead of the variable x is used to determine the identity of the object v-constructed by the construction ${}^2q\{x := {}^{12}q\}$ [replacement of the variable x by the single execution ${}^{12}q$ thus happens only "after" the variable x has been v-constructed by the variable q. Accordingly, the variable x merely determines the specific step of v-constructing when the single execution ${}^{12}q$ becomes relevant for v-constructing of the construction ${}^2q\{x := {}^{12}q\}$]. Consequently, the assumption of $q \rightarrow_v {}^{12}q$ is not required for the computation of the value of the construction ${}^2q\{x := {}^{12}q\}$, and the vicious cycle of $q \rightarrow_v {}^{12}q$ is thereby averted.

On this point, it is important to read carefully the definition *(iv) TPL: Def. v-constructing*, which states: "*If X is a construction and X v-constructs a construction Y, then ${}^2X$ v-constructs the object (if any) that is v-constructed by Y. Otherwise, ${}^2X$ does not v-construct anything.*" (Kosterec 2024, 168). It is the main and only rule for determining the identity of the object v-constructed by any double execution [including a double execution of the form ${}^2Y\{x := D\}$]. This rule explicitly and without exception requires that the construction Y [in Example 9, the single execution ${}^{12}q$] *must be* v-

c iff C v(d/x)-constructs c. (Duží et. al. 2010, 145; Kosterec 2024, 180). We can see that the compensation principle essentially states that if the construction D v-constructs the same object [D→$_v$d] as a certain variable x [i.e., under (v(d/x)], then the collisionless substitution of the construction D for this variable x [C(D/x)], should not alter the object v-constructed by construction C into which we substitute construction D for variable x [C(D/x)→$_v$c and also Cv(d/x)→$_v$c]. The compensation principle should be valid for any object d and c v-constructed by D and C, respectively.

constructed by the construction X [in Example 9, the variable q] in order to provide a value for an object v-constructed by double execution ${}^2X$ [in Example 9, the construction ${}^2q\{x := {}^{12}q\}$]. As I show above, this cannot be satisfied in the case of the construction ${}^2q\{x := {}^{12}q\}$. Therefore, although *the dynamic explanation* explains how to apply *(a) TPL: Def. collisionless substitution* in a way that avoids a vicious cycle, this explanation is not compatible with the *(iv) TPL: Def. v-constructing*. We may thus conclude that there is no rule in place in TPL that specifies the single execution ${}^{12}q$ as a source of the identity of an object v-constructed by ${}^2q\{x := {}^{12}q\}$, under the assumption that the replacement of the single execution ${}^{12}q$ for the variable x has occurred "after" the variable x is v-constructed by the variable q.

Therefore, neither of the proposed explanations satisfactorily explains the application of rule *(a) TPL: Def. collisionless substitution* in the case ${}^2q({}^{12}q/x) \leftrightarrow^2 q\{x := {}^{12}q\}$.

4. Conclusion

The definition of free occurrence of a variable in TPL permits collisionless substitution for occurrences of variables that are not (visible) parts of double execution. Rule *(a) TPL: Def. collisionless substitution* specifies that such collisionless substitution into the "hidden layers" of double execution yields a new construction of the form ${}^2Y\{x := D\}$.

The article shows that TPL provides neither a clear interpretation for the application of this rule nor a clear account of the structure of the resulting double execution. The article therefore proposes two possible interpretations for the application of the collisionless substitution in such cases, which labels *the static explanation* and *the dynamic explanation.*

The article shows that the collisionless substitution under *the static explanation* leads to an inconsistency that violates the compensation principle. By contrast, although *the dynamic explanation* explains how collisionless substitution may be applied in case of double execution without inconsistency, this explanation fails to conform to the general rule for v-constructing for the double execution. Consequently, the article concludes that TPL lacks clear rules for determining an object v-constructed by double

execution of the form $^2Y\{x := D\}$. This result poses a challenge for TPL, despite its potential as a promising new system of transparent logic.

References

Duží, Marie. 2012. "Towards an Extensional Calculus of Hyperintensions." *Organon F*, 19(1): 20–45.

Duží, Marie. 2014. "Structural Isomorphism of Meaning and Synonymy." *Computación y Sistemas*, 18(3): 439–53. https://doi.org/10.13053/cys-18-3-2018

Duží, Marie. and Číhalová, Martina. 2024. "Knowing Who Occupies an Office: Purely Contingent, Necessary and Impossible Offices." S*ynthese*, 203(6): 202. https://doi.org/10.1007/s11229-024-04596-x

Duží, Marie. and Jespersen, Bjørn. 2012. "Transparent Quantification into Hyperpropositional Contexts" de re"." *Logique et Analyse*, 55(220): 513–54.

Duží, Marie and Jespersenn, Bjørn. 2025. "Twin Procedures, Two Kinds of Variable Binding, and Two Kinds of Computation." *Annals of Pure and Applied Logic*, 177(4): 103693. https://doi.org/10.1016/j.apal.2025.103693

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

Duží Marie, Jespersen Bjørn and Glavaničová, Daniela. 2021. "Impossible individuals as necessarily empty individual concepts". In *Logic in High Definition. Trends in Logic*, (56): 177–202. Cham: Springer. https://doi.org/10.1007/978-3-030-53487-5_9

Jespersen, Bjørn and Duží, Marie. 2022. "Transparent Quantification into Hyperpropositional Attitudes de Dicto." *Linguistics and Philosophy*, 45(5): 1119–64. https://doi.org/10.1007/s10988-021-09344-9

Kosterec, Miloš. 2020. "Substitution contradiction, its resolution and the Church-Rosser Theorem in TIL." *Journal of Philosophical Logic*, 49(1): 121–33. https://doi.org/10.1007/s10992-019-09514-y

Kosterec, Miloš. 2021. "Substitution inconsistencies in Transparent Intensional Logic" *Journal of Applied Non-Classical Logics*, 31(3-4): 355–71. https://doi.org/10.1080/11663081.2021.1982553

Kosterec, Miloš. 2023. "A Hyperintensional Theory of (Empty) Names." *Erkenntnis*, 88(2): 511–29. https://doi.org/10.1007/s10670-020-00364-8

Kosterec, Miloš. 2024. *Transparent Logics. Small Differences with Huge Consequences*. Leiden: Brill. DOI: https://doi.org/10.1163/9789004703346

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

Raclavskỳ Jiří, Kuchyňka Petr and Pezlar, Ivo. 2015. *Transparentní intenzionální logika jako characteristica universalis a calculus ratiocinator*. Brno: Masarykova univerzita.

Tichỳ, Pavel. 1988. *The foundations of Frege's logic*. Berlin and New York: Walter de Gruyter. https://doi.org/10.1515/9783110849264