Raclavský, Jiří

Tichý's "Five modes of forming constructions"

Sborník prací Filozofické fakulty brněnské univerzity. B, Řada filozofická. 2000, vol. 49, iss. B47, pp. [75]-82

ISBN 80-210-2405-4
ISSN 0231-7664

Stable URL (handle): https://hdl.handle.net/11222.digilib/107196
Access Date: 21. 02. 2024
Version: 20220831

SBORNÍK PRACÍ FILOZOFICKÉ FAKULTY BRNĚNSKÉ UNIVERZITY
STUDIA MINORA FACULTATIS PHILOSOPHICAE UNIVERSITATIS BRUNENSIS
B 47, 2000

JIŘÍ RACLAVSKÝ

TICHÝ'S "FIVE MODES OF FORMING CONSTRUCTIONS"

Pavlu Materovi k 70. narozeninám

Tichý's (original and final) definitions of construction in Tichý 1988 are presented and commented. It is necessary to say that these Tichý's definitions are embodied into the text of book another way than standard explanation is given. Especially, it is common to define first-order theory of types before exposing constructions. The present article arose for the seminary "Gottlob Frege's Foundation of Logic" as a presentation of respective paragraph 15, Chapter Five: A Hierarchy of Entities (Tichý 1998). Therefore the problem of variables (in Tichý's book paragraph 14) is mentioned rather briefly.

First, we should articulate one of several distinctions: we can distinguish between simple and compound constructions. (Convention: A construction containing a variable constructs one entity relative to one valuation and another entity relative to another. A construction constructs an entity relative to valuation v, we shall briefly say that a construction v-constructs the entity assigned to it by v.)

1) **simple constructions** ("atomic" construction – Materna 1998, p. 40; his tag is similar to λ-calculi, where x (if it is a variable) is an atomic term)

Variables are the only simple constructions. The variables ξ_n construct the n-th object from the given sequence of objects yielded by the valuation, for any valuation v the variable v-constructs what the valuation v assigns to it. A valuation is an objectual valuation; valuations are total functions that associate each variable with one object of the respective type; for every type α there are denumerably many (α-)variables at our disposal. The letters commonly used for variables (x, y, ...) are conceived to be names of variables here. Technically variables behave exactly like letters, but notice that the approach is strictly objectual.

2) **compound constructions** (Tichý 1988 s. 64; "molecular" constructions)

Constructions other than variables have constituent parts, we will discuss them after a while.

Further we will also distinguish between see, v-proper and v-improper constructions, and complete constructions and incomplete ones as well.

So we have to distinguish ***v*-proper** and ***v*-improper** construction: a construction which *v*-constructs nothing at all shall be called *v*-improper. Otherwise it is proper. There are three constructions which can be *v*-improper: execution, double execution and composition.

The latter distinction, **complete** and **incomplete construction** (we can also find the terms closed and open constructions; in Czech: úplné-neúplné, otevřené-uzavřené), is defined as follows: the incomplete construction is a construction containing at least one free variable. For example a variable (which is a construction) is a simple case of an incomplete construction. The complete constructions do not contain any free variable. Complete constructions construct independently of valuations. The definition of free or bound variables will be stated after explaining five modes of forming constructions.

Briefly: assignings of objects (of respective types) to variables are qualified by valuations.

Now, to define the class of constructions (which is infinite) inductively, we must specify the modes of forming constructions, i.e., of forming constructions from non-constructions („mere objects") and other constructions. In his book Tichý found useful to state all in all five such modes.

1. Trivialization (trivializace)

Trivialization X, symbolized ^{0}X, is a rudimentary construction.

Definition:

– Where X is any entity (any object or construction), we can consider trivial construction whose starting point, as well as outcome, is X itself. To realize, carry out, *trivialization* ^{0}X, we must start with X and leave it as it is. It constructs X without any change. No matter how complex the construction X itself may be, ^{0}X is quite trivial. Every construction can be trivialized.

(If X is a first order object, ^{0}X will be called a first order trivialization. There are also higher order trivializations – Materna 1998, p. 41.)

– Note that for no entity X and valuation *v* the trivialization ^{0}X is *v*-improper.

– Also note that what is *v*-constructed by ^{0}X never depends on *v*.

Examples: if X is a numerical construction, i.e., a construction which *v*-constructs numbers, then for any *v*, X *v*-constructs a number (if any), while ^{0}X *v*-constructs X; hence if *x* is a variable then ^0x *v*-constructs *x* for any *v*.

Another examples: ^{0}3 *v*-constructs 3. Analogously – applied to natural language – ^{0}Bill Clinton constructs the individual Bill Clinton (not the expression 'Bill Clinton', of course).

Further comments: Trivializations serve as "immediate" construction. They can be seen as one-step procedures. Their counterparts in the field of epistemology can be called "immediate identifications". Thus the trivialization is more important than it possibly seems: especially this construction enables us to distinguish between objects and the way they are constructed (objects are not con-

structions). The importance of trivialization will be obvious after introducing a ramified hierarchy of types.

In the first phase of development of TIL (viz. Tichý 1986, "Constructions") Tichý considered objects as trivial procedures (see Tichý 1996, "Konstrukce", p. 120, above; or on page 133 he wrote: Every object of type ξ is also ξ-construction; for any v it v-constructs itself). In this article Tichý also did not define the trivialization among constructions, the first occurrence of trivialization is in his book (Tichý 1988).

(We can pose the question whether trivialization is a simple construction. We cannot easily answer it. The trivialization is useful quite independently of such answer.)

2. Execution (provedení) (cited from Tichý 1988)

For any entity X we shall also speak of the *execution* of X and symbolize it as ^{1}X. (Currently it is symbolized only X.)

Definition:

– If X is a construction, ^{1}X is X. Construction consisting in executing construction X is none other than X itself. It v-constructs what is v-constructed by X.

– If, on the other hand, X is not a construction, then ^{1}X is the (abortive) construction whose starting point is X and which yields nothing, i.e., a non-construction cannot be executed. Thus if X is v-improper construction or not a construction at all, ^{1}X is v-improper.

Examples (x is a numerical variable):

^{1}3 is v-improper

^1x v-constructs the number assigned to x by v.

3. Double execution (dvojité provedení) (cited from Tichý 1988)

If what is constructed by X is itself a construction, one can execute X and go on and execute the result. This two-stage construction can be called *double execution* and symbolized as ^{2}X.

Definition:

– ^{2}X v-constructs what is v-constructed by what is v-constructed by X.

– For any entity X the construction ^{2}X is v-improper (i.e., yields, relative to v, nothing at all) if X is not itself a construction, or if it does not v-construct a construction, or if it v-constructs a v-improper construction.

Examples (x is a numerical variable):

a) ^2x is v-improper

b) $^2(^0x)$ v-constructs the same as x, i.e., the number assigned by v to x.

Note that ^{2}X is not the same as $^1(^1$X): if X is construction, ^{1}X is the same as X, thus $^1(^1$X) is the same as ^{1}X which in turn is X. Hence, if c is a variable

ranging over numerical constructions, ${}^0c$, ${}^1c$, a ${}^2c$ are three distinct constructions. ${}^0c$ constructs c, quite independently of v. ${}^1c$ v-constructs numerical construction that is assigned by v to c. ${}^2c$ v-constructs whatever number (if any) which is v-constructed by construction which v assigns to c (note that what is v-constructed by ${}^2c$ may depend on what v assigns to variables others than c).

Remark: It is easy to see that inductively the whole class of executions can be defined (Materna 1998, p. 39).

Another remark: The last two constructions, viz. execution and double execution, can be found only in Tichý 1988. Other research workers of TIL do not use these constructions – Materna (in 1998) thinks that they are not principally necessary for logical analysis of natural language (except special cases; execution can be bypassed by the function from a construction to what it constructs).

4. Composition (kompozice, složení)

Let F be a construction of a mapping and X a construction of an argument of the mapping. F and X can be combined into a compound construction which consists in i) executing F (remember foregoing definition of execution), thus obtaining a mapping, then ii) executing X, thus obtaining an argument of the mapping, and then iii) applying the mapping to the argument, thus obtaining the value (if any) of the former at the latter. We shall call this compound construction the 'composition' F and X, or briefly [FX] . (Surely, the symbol '[FX] ' names the construction, not the number constructed by it.) This kind of constructions is very similar to the application of λ-calculi, where the λ-term [M A] means application of the mapping M to the argument A. Now to generalize the above we can put X_0 to equal F and $X_1 \ldots X_m$ to equal X (argument which can be, of course, m-tuple).

Definition:

– Let X_0, X_1, ..., X_m be arbitrary constructions. By the *composition* $[X_0X_1 \ldots X_m]$ *of* constructions X_0, X_1, ..., X_m (in this order) we shall understand the construction consisting in: i) executing X_0 to obtain an m-ary mapping, then ii) executing X_1, ..., X_m to obtain an m-tuple of entities, and then iii) applying that mapping to the m-tuple.

– Thus for any valuation v, $[X_0X_1 \ldots X_m]$ is v-improper, if i) one of constructions $X_1, \ldots X_m$ is v-improper, or if ii) X_0 does not v-construct a mapping which is defined at the m-tuple of entities v-constructed by X_1, ..., X_m. (After introducing the type theory we can consider the composition of type-incompatible entities also as improper.)

If X_0 does construct such a mapping then $[X_0X_1 \ldots X_m]$ v-constructs the value which the mapping takes at the m-tuple.

Remember also that X_0, X_1 ..., X_m can be complete or incomplete constructions.

Attention: The square brackets are not the same brackets as the brackets used for the denoting of m-tuples (by the way, Tichý disliked m-tuples, he construed them only as an aid for an abbreviation of realizing m-ary functions (the functions applicable to the m-tuples of arguments); see Cmorej&Tichý 1998).

Another remark: In Tichý 1986 Tichý used Comp^m (F, X_1, ..., X_m) for composition, he also argued that it should be called composition, not application, because the parts are not lost here.

Example: let us start with 2+3, 2 and 3 are objects-numbers, + is the addition mapping; all objects must be trivialized, the resulting construction consists in applying the addition mapping to (the couple of) two and three: $[^0+\ ^02\ ^03]$. This construction constructs number 5 similarly like ^{05} but it is a distinct construction, these constructions are not identical only "congruent".

Another example: where × and – are the multiplication and the subtraction mappings, $[^0-\ [^0\times\ xx]\ ^03]$ is the incomplete construction of multiplying an unspecified number by itself and subtracting three from the result, the unspecified number will be given by a valuation.

Let us add the definition of congruency: Two constructions will be called *v*-**congruent** if they *v*-construct one and the same object or are both *v*-improper. Moreover, they will be called **congruent** if they are *v*-congruent for any *v*.

5. Closure (uzávěr)

A certain incomplete construction X (with an unspecified value of the variable included in X) can be turned into the complete construction of some mapping. We shall call this complete construction 'α-closure of [X] on x' and symbolized it $[\lambda_\alpha x\ [X]\ ]$. Thus if composition "computes" the value of some function at certain argument, the closure "generates" a function. The function arises as follows: we let run (every) variable through all valuations and hence the dependence on valuation is omitted. Note that notation '$[\lambda_\alpha x$ Y] ' names construction, not the mapping constructed by it. In closure we can recognize the third kind of term of λ-calculi: λ-abstraction (briefly abstraction; for a sign of λ we use the name λ-abstractor, briefly abstractor). Bottom index α shows the type of the values of the resulting mapping.

Definition:

– Let τ be a (certain) collection, x_1, ..., x_m distinct variables ranging over the respective collections ξ_1, ..., ξ_m and *v* a valuation. Any construction Y can be used in constructing a mapping from ξ_1, ..., ξ_m into τ ; we shall call this latter construction the *τ-closure Y on* x_1,..., x_m , or briefly $[\lambda_\tau x_1 \ldots x_m Y]$. For any valuation *v*, $[\lambda_\tau x_1 \ldots x_m Y]$ *v*-constructs the mapping which takes any X_1, ..., X_m of the respective types ξ_1, ..., ξ_m into that member (if any) of τ which is $v(X_1/x_1$, ..., $X_m/x_m)$-constructed by Y, where $v(X_1/x_1$, ..., $X_m/x_m)$ is like *v* except for assigning X_1 to x_1, ..., and X_m to x_m .

– Consequently, for any τ , Y, x_1, ..., x_m and *v*, construction $[\lambda_\tau x_1 \ldots x_m Y]$ is *v*-proper.

Remark: The valuation $v(X_1/x_1, \ldots, X_m/x_m)$, which could be called v', is such a valuation which is quite similar to valuation v, except assigning $X_1, \ldots, X_m$ to bound variables $x_1, \ldots, x_m$. In other words, the given valuation v is accepted only for variables (possibly occurring in Y) which are distinct from $x_1, \ldots, x_m$. Thus v does not concern $x_1, \ldots, x_m$ ($v(x_i)$ 1 $v'(x_i)$ but $v(y_i) = v'(y_i)$).

Another remark: Now when Y is v'-improper and we consequently cannot find the value for given argument, the function constructed by $[\lambda_\tau x_1 \ldots x_m Y]$ is undefined for this argument (the construction $[\lambda_\tau x_1 \ldots x_m Y]$ is, in spite of that, proper). Example of this: $[^0> x\ ^0 0]$ is improper but $\lambda x[^0> x\ ^0 0]$ constructs a function undefined at all arguments (in every line of the table).

Further comments: The construction called closure is the same as the functional instructions, prescriptions, (used commonly from the beginning of the 17th century) like $2x^2+3$. In the case of $2\times x^2+3$ and $y^2 \times y^2+3$, they are two distinct constructions of the same mapping. $2x^2+3$ can be naturally turned into the term of typed λ -calculi denoting the respective constructions. Whereas functions as mappings are 'flat' (Materna's term) – you cannot recognize the parts of Y, you can 'see' only a table with the m-tuples of arguments on the left side and the values on the right side, closures are structured: they might be construed as instructions how to create a function – you can see every step, every partial instruction.

Remark: All the time Tichý used the term collection because he dismissed the term set which could be possibly problematized by an alternative approach to the theory of set.

(Remark: in Tichý 1986 the closure is signed $\mathrm{Clos}^{x1,\ldots xm}(Y)$.)

Remember also carefully the difference between construction and expression. The term $[\lambda x_1 x_2\ [^0+ x_1\ x_2]]$ contains four occurrences of variables but the construction only two occurences of variables. And, moreover: no construction contains λ .

Examples (x is a numerical variable)*:*

a) the mathematical expression f: (x_1+x_2) can be transcribed as follows: $[\lambda\ x_1 x_2\ [^0+ x_1\ x_2]]$

b) incomplete construction $[^0+ x_1\ ^0 3]$ can be turned into the complete construction $[\lambda x_1\ [^0+ x_1\ ^0 3]\ ]$ (commonly written x_1+3)

Convention: we can omit the outermost brackets: so $[\lambda x_1 x_2\ [^0+ x_1\ x_2]$ can be written $\lambda x_1 x_2 [^0+ x_1\ x_2]$. Also $[\lambda x_1 [\lambda x_2 \ldots\ [\lambda x_m\ Y] \ldots]]$ can be abbreviated by $\lambda\ x_1 \lambda x_2 \ldots\ \lambda x_m\ Y$.

c) $\lambda\ x\ [^0> x\ ^0 0]$ constructs the function which assigns true to every number greater than 0, i.e., the class of positive numbers. (Remark concerning relations vs. functions (citing Materna 1998): Relations and functions are mutually convertible, i.e., any n-place relation can be viewed as an n-adic function, and any n-adic function is an n– or (n+1)-relation. Well, beginning with the notion of function enables us to define a most important operation – the application of a function to its arguments. In contrast to predicate logic which is based on relations, transparent intensional logic is based on functional approach.)

d) $\lambda x_1[^0> x_1\ x_2]$ v-constructs the function which assigns the truth value True to every number (x_1) if this number is greater than number assigned to x_2 by valuation. So when the valuation assigns x_2 number 0, the construction $\lambda\ x_1[^0> x_1\ x_2]$ v-constructs the class of positive numbers (without respect to any valuation for x_1). When the valuation assigns to x_2 number 6 then the construction $\lambda x_1[^0> x_1\ x_2]$ v-constructs the class of numbers greater than 6 (and again: we take into account all valuations v').

e) Now let be both variables from the previous example bound: $\lambda x_1 x_2\ [^0> x_1\ x_2]$, this construction constructs the function which associates every couple of numbers to a truth value quite independently of any valuation. This mean that $\lambda\ x_1 x_2\ [^0> x_1\ x_2]$ v-constructs the same as $^0>$, viz. the relation "greater than".

f) $\lambda x_1 x_2\ [^0> x_1\ x_2]$ constructs the function from couples of numbers to a truth value, the function that assigns true when the first number is greater than the second, whereas $\lambda x_1 x_2\ [^0> x_2\ x_1]$ constructs the relation < , because the relation is first applied to the second number.

Subconstructions (podkonstrukce)

It is also possible to add the definition of *subconstructions* which is not a part of Tichý's Chapter Five. (In λ-calculi the definition of subterms approximately corresponds to it.) For the definition see Materna 1998, p. 91 (or Materna 1995, p. 70).

Literature:

CMOREJ, Pavel & TICHÝ, Pavel (1998): Komplexy [Complexes]. *Organon F*, No. 2-3

MATERNA, Pavel (1989): *Logická analýza přirozeného jazyka* [Logical Analysis of Natural Language]. Praha: Academia

MATERNA, Pavel (1995): *Svět pojmů a logika* [The World of Concepts and Logic]. Praha: Filosofia AV ČR

MATERNA, Pavel (1998): *Concepts and Objects*. Helsinki: Acta Philosophica Fennica

TICHÝ, Pavel (1986): Constructions. Philosophy of Science 53, pp. 514-534; (in Czech: (1996): Konstrukce, in: O čem mluvíme? Vybrané stati k logice a sémantice, Praha: Filosofia)

TICHÝ, Pavel (1988): *The Foundations of Frege's Logic*. Berlin-New York: Walter de Gruyter

TICHÝ, Pavel (1995): Construction as the Subject Matter of Mathematics, In: W. Depauli-Schimanovich, E. Köhler, Fr. Stadler, eds.: *The Foundational Debate (Complexity and Constructivity in Mathematics and Physics)*, Dodrecht-Boston-London: Kluwer Academic Publishers, pp. 175-185; (in Czech: (1998) Konstrukce jako předmět matematiky, *Filosofický časopis 2*)

TICHÉHO „PĚT ZPŮSOBŮ FORMOVÁNÍ KONSTRUKCÍ"

Stať *Tichého „Pět způsobů formování konstrukcí"* podává, vysvětluje a komentuje Tichého definice z [Tichý 1988]. Konstrukce je v (Tichým vybudované) Transparentní intenzionální logice (TIL) explikací fregovského smyslu, toho, díky čemu výrazům rozumíme. Konstrukce konstruují intenze (resp. extenze), jsou to přitom nemnožinové entity. Tím je zachycen fakt, že např. pro jednu funkci existuje *n* funkčních předpisů (konstrukce jsou modifikací lambda termů). Identifikací (a rozčleněním druhů) konstrukcí se TIL zásadně odlišuje od jiných intenzionálních logik. Konstrukce nám umožňují lépe logicky analyzovat věty, a tudíž lépe podchytit vyplývání.