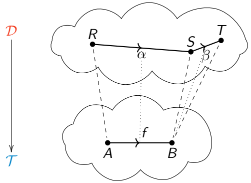
Type refinement systems and the categorical perspective on type theory

Noam Zeilberger

University of Birmingham

Workshop on Practical & Foundational Aspects of Type Theory
University of Kent
19 June 2018



What is type refinement?

Historically, *type refinement* is the name of a long-term project initiated by Frank Pfenning in the late 1980s, with the aims to

> *(a) capture more precise properties of programs that we already know to be well-typed in a simpler discipline in order to catch more errors, and (b) retain the good theoretical and practical properties of the simpler disciplines such as effective type checking. (Pfenning, blog comment, 2015)*

The word "refinement type" has taken on a somewhat narrower meaning in some circles, but I will be using "type refinement" in an even broader sense than the original usage.[1]

[1] For additional background, see the notes to my OPLSS 2016 tutorial on "Principles of Type Refinement" (available on my webpage).

What is a type refinement system?

"Definition": a **type refinement system** is a type system built over a typed programming language, as an extra layer of typing.

What is a type system?

From *Types and Programming Languages* by B. Pierce (2002):

> *As with many terms shared by large communities, it is difficult to define "type system" in a way that covers its informal usage by programming language designers and implementors but is still specific enough to have any bite.*

What is a type?

Type theory is a bit unusual as a *mathematical theory* in that it does not define its supposed topic!

(Contrast: “group theory”, “knot theory”, “category theory”.)

Thesis

The reason why it is so hard to give a formal definition of “type” is that in practice the word covers two very different usages.

Intrinsic *vs.* Extrinsic

Reynolds called these the “intrinsic” and the “extrinsic” views.[2]

Also called “types à la Church” *vs.* “types à la Curry”.

Logical intuition:

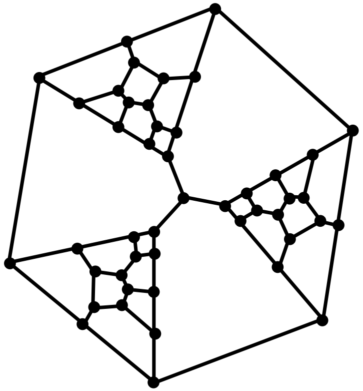
intrinsic type = domain of discourse

vs.

extrinsic type = predicate on domain of discourse

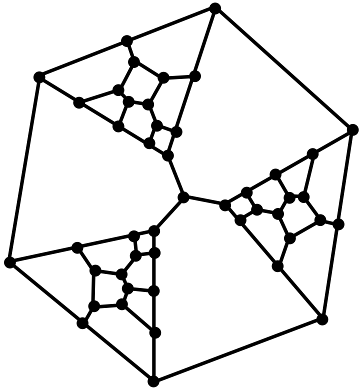
[2] John C. Reynolds (1998), *Theories of Programming Languages*.

Intrinsic *vs.* Extrinsic



: graph

Intrinsic *vs.* **Extrinsic**



: 4-colorable graph

The extrinsic view: Robin Milner (1978)

> *We now proceed, in outline, as follows. We define a new class of expressions which we shall call types; then we say what is meant by a value* **possessing** *a type. Some values have many types, and some have no type at all. In fact "wrong" has no type. But if a functional value has a type, then as long as it is applied to the right kind (type) of argument it will produce the right kind (type) of result-which cannot be "wrong"!*

(From "A Theory of Type Polymorphism in Programming".)

The intrinsic view: Dana Scott (1980)

> *General category theory is a very pure theory: it is the milk-and-water theory of functions under composition. This composition operation is associative and possesses neutral elements (compositions of zero terms). That is about all you can say about it except to stress that it is also a rather bland theory of types.* **Every function f has a (unique) domain and codomain,** *and we write $f : \operatorname{dom} f \to \operatorname{cod} f$.*

(From "Relating theories of the λ-calculus".)

Problem: the naive reading of type theory through the lens of category theory is biased towards the <u>intrinsic</u> view of typing, yet the <u>extrinsic</u> view is of fundamental importance in practice.

The naive reading of TT via CT

type system $\rightsquigarrow$ category of well-typed terms

$$x : A \vdash t : B \qquad \rightsquigarrow \qquad [\![A]\!] \xrightarrow{[\![t]\!]} [\![B]\!]$$

Subtyping and polymorphism

The naive reading makes it difficult to interpret typing rules such as **subsumption** or **intersection introduction**, which assign one term multiple types:

$$\frac{\Gamma \vdash t : A \quad A \leq B}{\Gamma \vdash t : B} \qquad \frac{\Gamma \vdash t : A \quad \Gamma \vdash t : B}{\Gamma \vdash t : A \wedge B}$$

Indeed, it is *ungrammatical* for a morphism in a category to have two different codomains. . .

$$* \quad \begin{array}{l} [\![\Gamma]\!] \xrightarrow{[\![t]\!]} [\![A]\!] \\ [\![\Gamma]\!] \xrightarrow{[\![t]\!]} [\![B]\!] \end{array}$$

Type inference and principal types

The naive reading also fails to give much insight into the concept of the **principal type** (= most general type) of a term.

$$\lambda x.\lambda y.\lambda z.x(yz) : (bool \to bool) \to (bool \to bool) \to (bool \to bool)$$
$$\lambda x.\lambda y.\lambda z.x(yz) : (int \to bool) \to (bool \to int) \to (bool \to bool)$$
$$\lambda x.\lambda y.\lambda z.x(yz) : (B \to C) \to (A \to B) \to (A \to C) \quad [\text{for any } A, B, C]$$

Missing players

More fundamentally, in limiting to "well-typed terms", the naive reading of TT through the lens of CT collapses the distinction between terms, typing judgments, and typing derivations.

Coherence theorems: replacing naivete with subtlety

A sophisticated resolution of this problem is to instead define the semantics of a language *by induction on typing derivations*, and then prove a **coherence theorem** that the meaning of a judgment is independent of its derivation:

$$\text{If } \Gamma \vdash \overset{\alpha_1}{t} : A \text{ and } \Gamma \vdash \overset{\alpha_2}{t} : A \text{ then } [\![\alpha_1]\!] = [\![\alpha_2]\!] : [\![\Gamma]\!] \to [\![A]\!]$$

In general, the proof of coherence is non-trivial...

This approach has been nicely discussed by Reynolds.[3]

[3] See especially: "The Meaning of Types: from Intrinsic to Extrinsic Semantics" (2000).

Our goal: unify the intrinsic and the extrinsic views of typing, and place the latter on a solid categorical footing.

Approach: stay naive (rather than subtle), just not *too* naive!

Reading a functor as a type refinement system

Refinement systems

$\boxed{\textbf{Idea:}\ \text{any functor } p : \mathcal{D} \to \mathcal{T} \text{ can be seen as a proof system.}}$

For objects $R \in \mathcal{D}$ and $A \in \mathcal{T}$, we write $R \sqsubset A$ and say that

R is a refinement of A

if $p(R) = A$.

Definition: A **typing judgment** is a triple (R, f, S)

$$(\text{written } R \underset{f}{\Longrightarrow} S)$$

consisting of a morphism $f : A \to B$ (in $\mathcal{T}$) together with a pair of refinements $R \sqsubset A$ and $S \sqsubset B$. A **derivation** of a judgment (R, f, S) is a morphism $\alpha : R \to S$ (in $\mathcal{D}$) such that $p(\alpha) = f$.

Refinement systems

The basic *typing rules* are composition and identity:

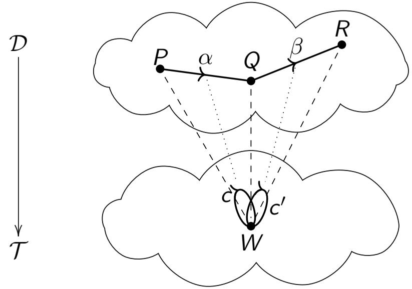
$$\frac{R \underset{f}{\Longrightarrow} S \quad S \underset{g}{\Longrightarrow} T}{R \underset{f;g}{\Longrightarrow} T} \qquad \frac{}{R \underset{\mathrm{id}}{\Longrightarrow} R}$$

The way to read a rule is: *given derivations of the premises, we can construct a derivation of the conclusion.*[4]

[4]We assume implicitly that judgments are well-formed. The composition and identity rules thus correspond to functoriality of $p : \mathcal{D} \to \mathcal{T}$.

A motivating example: Hoare logic

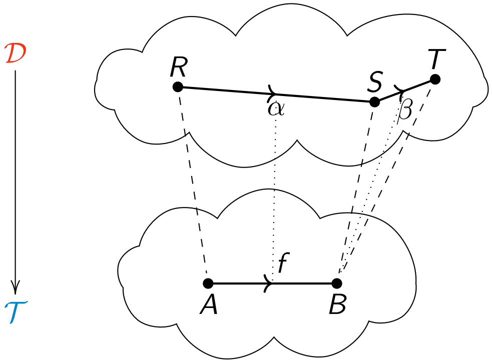
Take $\mathcal{T}$ as a one-object category of *state transformers*, $\mathcal{D}$ as a category of *state predicates* and *verified transitions*.



There is a derivation of (P, c, Q) just in case c takes any state satisfying P to a state satisfying Q (written "$\{P\}c\{Q\}$"). In particular, (P, id, Q) is derivable just in case P entails Q.

Intuition: type refinement system = "logic + side-effects".

More general picture



$$R \overset{\alpha}{\underset{f}{\Longrightarrow}} S \quad S \overset{\beta}{\leq} T$$

Important: Distinction between *refinement* and *subtyping*.

A straightforward interpretation of subsumption

Exercise: Show that the rules of (covariant and contravariant) subsumption are valid.

$$\frac{R \underset{f}{\Longrightarrow} S \quad S \leq T}{R \underset{f}{\Longrightarrow} T} \qquad \frac{R \leq S \quad S \underset{g}{\Longrightarrow} T}{R \underset{g}{\Longrightarrow} T}$$

The logic of monoidal closed bifibrations

Type refinement and monoidal closed bifibrations

If we view a functor as a type refinement system, it is natural to consider when it is both (symmetric or cartesian) **monoidal closed** and a **(bi)fibration**.

Paul-André Melliès and I have explored this for a number of years:

- Type refinement and monoidal closed bifibrations. arXiv:1310.0263
- Functors are type refinement systems. POPL 2015.
- An Isbell duality theorem for type refinement systems. MSCS.
- A bifib'l reconst. of Lawvere's presheaf hyperdoctrine. LICS 2016.

Broadly, the idea goes back to Lawvere's "hyperdoctrines" (1969), but with conceptual and technical differences. Our approach is closely related to work by Hermida (1993), Hasegawa (1999), Katsumata (2005), Atkey et al. (2011), and others.

Monoidal closed refinement systems $(I, \otimes, \multimap)$

A **monoidal closed refinement system** is a strict monoidal closed functor $p : \mathcal{D} \to \mathcal{T}$ between monoidal closed categories.

(Often assumed symmetric $A \otimes B \cong B \otimes A$ or cartesian $A \otimes B \cong A \times B$.)

Monoidal closed refinement systems $(I, \otimes, \multimap)$

Explicitly, a monoidal closed refinement system is a functor admitting the following refinement rules:

$$\frac{}{I \sqsubset I} \quad \frac{R \sqsubset A \quad S \sqsubset B}{R \otimes S \sqsubset A \otimes B} \qquad \frac{R \sqsubset A \quad S \sqsubset B}{R \multimap S \sqsubset A \multimap B}$$

and typing rules:

$$\frac{}{I \underset{I}{\Longrightarrow} I} \quad \frac{R_1 \underset{f}{\Longrightarrow} R_2 \quad S_1 \underset{g}{\Longrightarrow} S_2}{R_1 \otimes S_1 \underset{f \otimes g}{\Longrightarrow} R_2 \otimes S_2} \quad \dfrac{R \otimes S \underset{f}{\Longrightarrow} T}{\overline{S \underset{curry(f)}{\Longrightarrow} R \multimap T}}$$

and satisfying some equations.

(Some subtyping rules are also a consequence:)

$$\frac{}{I \leq I} \quad \frac{R_1 \leq R_2 \quad S_1 \leq S_2}{R_1 \otimes S_1 \leq R_2 \otimes S_2} \qquad \frac{R_2 \leq R_1 \quad S_1 \leq S_2}{R_1 \multimap S_1 \leq R_2 \multimap S_2}$$

Bifibrations ($\mathsf{push}_f, \mathsf{pull}_f$)

A **bifibration** is a functor $p : \mathcal{D} \to \mathcal{T}$ that admits both *left-cartesian liftings* and *right-cartesian liftings* of all maps.

In the language of type refinement, this is equivalent to the existence of operations:

$$\frac{R \sqsubset A \quad f : A \to B}{\mathsf{push}_f\, R \sqsubset B} \qquad \frac{f : A \to B \quad S \sqsubset B}{\mathsf{pull}_f\, S \sqsubset A}$$

such that there is a one-to-one correspondence of derivations:

$$\frac{R \underset{f;g}{\Longrightarrow} R'}{\overline{\mathsf{push}_f\, R \underset{g}{\Longrightarrow} R'}} \qquad \frac{S' \underset{e;f}{\Longrightarrow} S}{\overline{S' \underset{e}{\Longrightarrow} \mathsf{pull}_f\, S}}$$

(In particular, canonical derivations of $R \underset{f}{\Longrightarrow} \mathsf{push}_f\, R$ and $\mathsf{pull}_f\, S \underset{f}{\Longrightarrow} S$.)

Bifibrations $(\mathsf{push}_f, \mathsf{pull}_f)$

In the example of Hoare logic, a pushforward of a predicate along a command corresponds to a *strongest postcondition*, and dually, a pullback corresponds to a *weakest precondition*:

$$\frac{\{P\}c\{Q\}}{\overline{sp(c,P) \vDash Q}} \qquad \frac{\{P\}c\{Q\}}{\overline{P \vDash wp(c,Q)}}$$

Note that such strongest postconditions and weakest preconditions need not always exist, depending on the language of predicates and commands. In other words, the associated refinement system $p : \mathcal{D} \to \mathcal{T}$ is $\underline{\text{not}}$ necessarily a bifibration (although we can try to interpret it in one).

Monoidal closed bifibrations $(I, \otimes, \multimap, \mathsf{push}_f, \mathsf{pull}_f)$

Definition: **mc bifibration** = [monoidal closed + bifibration]

We therefore have a family of adjunctions:

$$\begin{tikzcd}
\mathcal{D} \arrow[r, bend left, "R\otimes-"] \arrow[r, phantom, "\perp"] \arrow[d] & \mathcal{D} \arrow[l, bend left, "R\multimap-"] \arrow[d] \\
\mathcal{T} \arrow[r, bend left, "A\otimes-"] \arrow[r, phantom, "\perp"] & \mathcal{T} \arrow[l, bend left, "A\multimap-"]
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{D}_A \arrow[r, bend left, "\mathsf{push}_f"] \arrow[r, phantom, "\perp"] & \mathcal{D}_B \arrow[l, bend left, "\mathsf{pull}_f"]
\end{tikzcd}$$

and some distributivity principles hold automatically:

$$\mathsf{push}_{(f\otimes g)}(R \otimes S) \equiv \mathsf{push}_f\, R \otimes \mathsf{push}_g\, S \qquad (1)$$

$$\mathsf{push}_f\, R \multimap \mathsf{pull}_g\, S \equiv \mathsf{pull}_{(f\multimap g)}(R \multimap S) \qquad (2)$$

Monoidal closed bifibrations $(I, \otimes, \multimap, \mathsf{push}_f, \mathsf{pull}_f)$

Another interesting feature of *symmetric* monoidal closed bifibrations is that they give rise to *two* kinds of "trialities":

$$\dfrac{\dfrac{R \otimes S \underset{f}{\Longrightarrow} T}{S \underset{curry(f)}{\Longrightarrow} R \multimap T}}{R \underset{rcurry(f)}{\Longrightarrow} S \multimap T} \qquad\qquad \dfrac{\dfrac{R \underset{f}{\Longrightarrow} S}{\mathsf{push}_f\, R \le S}}{f \le \mathsf{pull}_f\, S}$$

where *right-currying* is defined by symmetry and (left-)currying.

Monoidal closed bifibrations $(I, \otimes, \multimap, \mathsf{push}_f, \mathsf{pull}_f)$

$\mathrm{SubSet} \to \mathrm{Set}$ is a basic example of a (cartesian) mc bifibration:

$$\begin{aligned}
(R \subseteq A) \times (S \subseteq B) &= (\{\, (a,b) \mid a \in R \wedge b \in S \,\} \subseteq A \times B) \\
(R \subseteq A) \to (S \subseteq B) &= (\{\, f \mid a \in R \Rightarrow f(a) \in S \,\} \subseteq A \to B) \\
\mathsf{push}_f(R \subseteq A) &= (\{\, f(a) \mid a \in R \,\} \subseteq B) \\
\mathsf{pull}_f(S \subseteq B) &= (\{\, a \mid f(a) \in S \,\} \subseteq A)
\end{aligned}$$

$\mathrm{Rel}_\bullet \to \mathrm{Rel}$ gives a more interesting (non-cartesian) example:

$$\begin{aligned}
(R \subseteq A) \otimes (S \subseteq B) &= (\{\, (a,b) \mid a \in R \wedge b \in S \,\} \subseteq A \times B) \\
(R \subseteq A) \multimap (S \subseteq B) &= (\{\, (a,b) \mid a \in R \Rightarrow b \in S \,\} \subseteq A \times B) \\
\mathsf{push}_f(R \subseteq A) &= (\{\, b \mid \exists a.\, a[f]b \wedge a \in R \,\} \subseteq B) \\
\mathsf{pull}_f(S \subseteq B) &= (\{\, a \mid \forall b.\, a[f]b \Rightarrow b \in S \,\} \subseteq A)
\end{aligned}$$

(Replacing relations by distributors and subsets by presheaves yields a yet more interesting and representative mc bifibration $\mathrm{Dist}_\bullet \to \mathrm{Dist}$.)

Logic inside a monoidal closed bifibration

One of the original motivations for my work with Paul-André was the idea of using the logical connectives $\otimes / \multimap$ and push_f / pull_f to *reason about* formal systems, in the style of a "logical framework".

This idea is also closely related to recent work by Licata, Shulman, and Riley presented at FSCD 2017.

Example: the bifibrational Day construction

Proposition. Let $p : \mathcal{D} \to \mathcal{T}$ be a mc bifibration. Any monoid $(A, m : A \otimes A \to A, e : 1 \to A)$ in $\mathcal{T}$ determines a mc structure on the fiber $\mathcal{D}_A$, with unit, tensor and implication defined by:

$$I_A \overset{\text{def}}{=} \mathsf{push}_e\, I$$

$$R \otimes_A S \overset{\text{def}}{=} \mathsf{push}_m(R \otimes S)$$

$$R \multimap_A S \overset{\text{def}}{=} \mathsf{pull}_{curry(m)}(R \multimap S)$$

(For example, in the relational model this unwinds to:)

$$\begin{aligned} a \in I_A &\iff [e]a \\ a \in R \otimes_A S &\iff \exists a_1, a_2.\, (a_1, a_2)[m]a \wedge a_1 \in R \wedge a_2 \in S \\ a \in R \multimap_A S &\iff \forall a_1, a'.\, (a_1, a)[m]a' \Rightarrow a_1 \in R \Rightarrow a' \in S \end{aligned}$$

Example: fibrational biorthogonality

Proposition. Let $p : \mathcal{D} \to \mathcal{T}$ be a symmetric mc fibration. Any binary operation $m : A \otimes B \to C$ equipped with a refinement $T \sqsubset C$ of its codomain induces a contravariant adjunction between the fibers $\mathcal{D}_A$ and $\mathcal{D}_B$, defined by "negation into T relative to m":

$$R^{\perp} \overset{\mathrm{def}}{=} \mathsf{pull}_{curry(m)}\,(R \multimap T) \qquad (R \sqsubset A)$$

$$^{\perp}S \overset{\mathrm{def}}{=} \mathsf{pull}_{rcurry(m)}\,(S \multimap T) \qquad (S \sqsubset B)$$

(For example, in the relational model this unwinds to:

$$b \in R^{\perp} \iff \forall a.\,(a,b)[m]c \Rightarrow a \in R \Rightarrow c \in T$$

$$a \in {}^{\perp}S \iff \forall b.\,(a,b)[m]c \Rightarrow b \in S \Rightarrow c \in T$$

E.g., when $m = \mathrm{id}$ and $T =$ anti-diagonal on A, then $R^{\perp} = {}^{\perp}R = A \setminus R$.)

Example: a naive take on proof-theoretic semantics

Suppose we want to interpret a logical formula ϕ as the subset

$$\phi^+ = \{\, \Gamma \mid \Gamma \vdash \phi \,\}$$

of contexts in which it is derivable, or alternatively as the presheaf

$$\phi^+ = \Gamma \mapsto \{\, \alpha \mid \Gamma \overset{\alpha}{\vdash} \phi \,\}$$

sending a context to the set of proofs of ϕ in that context.

More abstractly, we can think of ϕ as a *refinement*

$$\phi^+ \sqsubset W$$

of a **type of contexts** W, and then try to interpret it in a concrete model such as $\mathrm{SubSet} \to \mathrm{Set}$ or $\mathrm{Rel}_\bullet \to \mathrm{Rel}$ or $\mathrm{Dist}_\bullet \to \mathrm{Dist}$.

Example: a naive take on proof-theoretic semantics

Consider the connectives $\otimes$ and $\&$ of linear logic. The right rules

$$\frac{\Gamma_1 \vdash \phi \quad \Gamma_2 \vdash \psi}{\Gamma_1, \Gamma_2 \vdash \phi \otimes \psi}\ \otimes R \qquad \frac{\Gamma \vdash \phi \quad \Gamma \vdash \psi}{\Gamma \vdash \phi \mathbin{\&} \psi}\ \& R$$

should now be interpreted as derivations

$$\phi^+ \otimes \psi^+ \overset{\otimes R}{\underset{m}{\Longrightarrow}} (\phi \otimes \psi)^+ \qquad \phi^+ \otimes \psi^+ \overset{\& R}{\underset{j}{\Longrightarrow}} (\phi \mathbin{\&} \psi)^+$$

where $m : W \otimes W \to W$ stands for the operation of concatenating contexts and $j : W \otimes W \to W$ for the (partial) operation of joining identical contexts (dual to duplication $d : W \to W \otimes W$).

Idea: $\otimes R$ and $\& R$ only differ in “side-effects” on the context.

Example: a naive take on proof-theoretic semantics

Then we have that

$$\mathsf{push}_j(\phi^+ \otimes \psi^+) \equiv \mathsf{pull}_d(\phi^+ \otimes \psi^+) \equiv (\phi \mathbin{\&} \psi)^+$$

since the $\& R$ rule is *invertible*, in the sense that any proof of $\Gamma \vdash \phi \mathbin{\&} \psi$ may be factored as a pair of proofs of $\Gamma \vdash \phi$ and $\Gamma \vdash \psi$.

On the other hand, the subtyping entailment

$$\mathsf{push}_m(\phi^+ \otimes \psi^+) \leq (\phi \otimes \psi)^+$$

is *strict*, in the sense that a proof of $\Gamma \vdash \phi \otimes \psi$ does not necessarily factor into a pair of proofs of $\Gamma_1 \vdash \phi$ and $\Gamma_2 \vdash \psi$, for some Γ_1 and Γ_2 whose concatenation is Γ. (Consider $\phi \otimes \psi \vdash \phi \otimes \psi$.)

This is one manifestation of *polarity* in linear logic.

Perspectives

A few ideas:

1. The extrinsic view of types is important to the practice of type theory, since it plays a role in fundamental concepts such as subtyping, polymorphism, and type inference.
2. Contrary to common dogma, both intrinsic and extrinsic views are compatible with a categorical perspective on type theory, by treating type systems as functors rather than as categories.
3. When we interpret a type refinement system in a mc bifibration, the interplay between the connectives $\otimes / \multimap$ and $\mathsf{push}_f \,/\, \mathsf{pull}_f$ gives rise to particularly rich logical phenomena.
4. Type refinement provides a natural bridge between reasoning about stateful computations and reasoning about contexts in substructural logic.

A few questions and a claim:

1. Functors compose, so the refinement relation can be iterated. Does this lead to interesting type refinement hierarchies?
2. Should the refinement relation really be modelled as a functor, or as a more general distributor?
3. It is possible to generalize Lawvere's definition of equality in hyperdoctrines to model *directed* identity predicates in mc bifibrations (LICS 2016). How does this tie with HoTT?
4. More generally, how should the study of type refinement systems influence our view of dependent type theory?
5. Type theory is ripe to be developed as an *axiomatic theory*, but to do so successfully it cannot overlook the miraculous existence of terms independently of their types.

Postscript: Louis Kauffman (2001)

> *The reason, I believe, that portmanteau and pivot are so important to find in looking at formal systems, and in particular symbolic logic, is that the very attempt to make formal languages is fraught with the desire that each term shall have a single well assigned meaning. It cannot be! The single well-assigned meaning is against the nature of language itself. All the formal system can actually do is choose a line of development that calls some entities elementary (they are not) and builds other entities from them. Eventually meanings and full relationships to ordinary language emerge.*

(From “The Mathematics of Charles Sanders Peirce”.)