



More Than One Way to Go On: *Reference, Function, and the Typing of Engineered Concepts*

Mais de uma maneira de continuar: referência, função e a tipagem de conceitos engenheirados

Author: Bernardo Alonso

ORCID: [0000-0003-3595-4907](https://orcid.org/0000-0003-3595-4907)

Primary contact:

bernardo.alonso@ufmt.br

Affiliation: Department of Philosophy, [Federal University of Mato Grosso](https://www.ufmt.br), Cuiabá-MT, Brazil.

How to cite

ALONSO, B. More Than One Way to Go On: Reference, Function, and the Typing of Engineered Concepts. *Sofia*, Espírito Santo, Brasil, v. 15, n. 2, p. e15254178, 2026. DOI: <https://doi.org/10.47456/sofia.v15i2.54178> Disponível em : <https://periodicos.ufes.br/sofia/article/view/54178> . Acesso em: 20 out. 2026.

Abstract

Conceptual engineering is often described as the practice of fixing the reference of our terms. This article argues that the description misplaces both the source of the practice and the site of its assessment. The reason for revising a concept, wherever the literature supplies one, is functional, and the functional rationale is a debt to pragmatism, from the later Wittgenstein through Sellars and Brandom to Price. The pragmatist then owes an account of how revisions are assessed once reference is set aside, and the article supplies one by translation. Where Carnap and Montague translated natural into artificial language, the proposal translates normative role into functional type. A concept becomes an interface with laws, a way of using it an instance that satisfies the laws, and a revision becomes the replacement of one instance by another beneath a shared interface. Worked in Haskell and Julia and read through the propositions-as-types correspondence, the translation makes the pragmatist slogan that there is always more than one way to go on a checkable property of a typed language. Reference survives as one function among several that an interface may demand, and the metasemantic category earns its place only where a concept's job includes tracking.

Keywords: conceptual engineering; neopragmatism; reference; inferentialism; type classes; propositions as types.

Resumo

A engenharia conceitual é frequentemente descrita como a prática de fixar a referência de nossos termos. Este artigo sustenta que essa descrição erra quanto à origem da prática e ao lugar de sua avaliação. A razão para revisar um conceito, onde quer que a literatura ofereça uma, é funcional, e a razão funcional é uma dívida com o pragmatismo, do segundo Wittgenstein, passando por Sellars e Brandom, até Price. O pragmatista deve então explicar como avaliar revisões sem a referência, e o artigo o faz por tradução. Onde Carnap e Montague traduziram a linguagem natural em linguagem artificial, a proposta traduz papel normativo em tipo funcional. Um conceito torna-se uma interface com leis, um modo de usá-lo torna-se uma instância que satisfaz as leis, e uma revisão torna-se a substituição de uma instância por outra sob uma interface comum. Elaborada em Haskell e Julia, e lida pela correspondência proposições-como-tipos, a tradução converte o lema pragmatista de que há sempre mais de uma maneira de continuar em uma propriedade verificável de uma linguagem tipada. A referência sobrevive como uma função entre outras que uma interface pode exigir, e a categoria metasemântica só se justifica para conceitos cuja tarefa inclui rastrear.

Keywords: engenharia conceitual; neopragmatismo; referência; inferencialismo; classes de tipos; proposições como tipos.

Recebido: 04/09/2026

Received: 04/09/2026

Aprovado: 13/10/2026

Approved: 13/10/2026

Publicado: 20/10/2026

Published: 20/10/2026

1. Introduction

The literature on conceptual engineering has a favourite verb, and the verb is fix. Cappelen's (2018) book is called *Fixing Language*, and the title carries both senses of the word at once: our concepts are defective and want repair, and repair consists in fixing, in the sense of settling, what they pick out. The double sense is not an accident of the title. It expresses a picture on which the object of engineering is a reference relation, the defect in a concept is a defect in what it refers to or in how it does so, and the success of a project is measured by whether the relation has been moved to where the engineer wanted it. Haslanger's (2000) proposal to define woman in terms of subordinate social position is read, on this picture, as a proposal to refix the extension of a word. The International Astronomical Union's (2006) reclassification of Pluto is read as a decision about what planet refers to. The externalist challenge that Cappelen raises against his own project, that reference is fixed by factors outside any speaker's control and so cannot be engineered by fiat, is a challenge that has force only if reference is what is being engineered.

This article argues that the picture gets the practice wrong in a way that matters for its assessment, and it proposes a replacement that can be checked rather than merely asserted. The argument has two halves, and I will state them before defending them. The first half is that conceptual engineering, in those strands of it that supply a reason for revising anything, is pragmatist in its foundations whether or not its practitioners say so. The reason one gives for wanting a better concept of woman or planet or knowledge is never that the old reference relation was wrong, since reference relations are not the sort of thing that can be wrong. The reason is that the concept was doing a job badly. Doing a job is a functional notion, and the account of language on which words are tools that do jobs, and on which the space in which they do them is a space of commitments and entitlements rather than of word-world hooks, is the account that runs from the later Wittgenstein through Sellars and Brandom to Price. The second half is that a pragmatist owes an account of assessment. If there is, as the pragmatist insists, always more than one way to go on, the engineer needs some way of saying which ways are better, and the standing complaint that pragmatism leaves conceptual choice arbitrary (Thomasson, 2020) has to be met with something firmer than reassurance.

I meet it by translation. Where Carnap (1950) translated natural into artificial language and called the result explication, and where Montague (1973) translated fragments of English into a typed lambda calculus and called the result semantics, I translate normative role into

functional type. The translation takes a concept to be an interface that states what any way of using it must provide, takes a particular way of using it to be an instance that provides it, and takes a revision to be the swapping of instances beneath a fixed interface. The reason for carrying out the translation in a strongly typed functional language rather than in prose is that the language checks what prose can only assert. That there is more than one way to go on becomes the fact that an interface admits more than one instance. That some ways are better than others becomes the fact that instances can be tested against the laws the interface declares. And the question in my title, whether conceptual engineering needs the metasemantic category of reference, receives an answer with a shape rather than a verdict. Reference is one function an interface may demand, on a par with inferential and expressive functions, and the engineer needs it exactly when the concept under revision has tracking among its jobs.

Section 2 sets out the pragmatist inheritance and argues that the functional turn in recent conceptual engineering is that inheritance made explicit. Section 3 examines the reference-fixing picture and the objections it has generated, and argues that the objections share the picture's assumption. Section 4 builds the translation, first in Haskell, then in Julia, and then reads it through the propositions-as-types correspondence. Section 5 draws the consequence for reference. Section 6 answers objections, and Section 7 concludes.

2. The Pragmatist Inheritance

To ask to what extent the idea of conceptual engineering rests on pragmatist views of language, one has to say which pragmatism and which idea. The pragmatism I have in mind is the one Price (2011, 2013) has spent two decades assembling out of parts that were lying around separately: the later Wittgenstein's insistence that the meaning of a word is its use in a practice, Sellars's (1954, 1956) relocation of conceptual content from the relation between a sign and a thing to the position of the sign in a game of moves and countermoves, and Brandom's (1994) systematisation of that position as a structure of commitments and entitlements. The common thread, in Price's telling, is a refusal to take the representational idiom as the default account of what a vocabulary is for. Some vocabularies track features of the environment, and for those the idiom of reference and truth-conditions is apt in a way Price is willing to call *e*-representation. But other vocabularies, the modal, the normative, the probabilistic, the semantic itself, do something else, and the mistake of what Price calls representationalism is to assume that they must all be doing the first thing badly because they

are not obviously doing it well. The alternative is functional pluralism: vocabularies differ in what they are for, the question to ask of any of them is what job it does in the life of creatures like us, and the representational job is one job among several rather than the standard against which the rest are measured.¹

The normative dimension of this picture comes from Sellars and Brandom rather than from Price, and it is the dimension on which the argument of this article turns. Sellars (1956, sec. 36) distinguished the space of reasons from the space of causes and placed conceptual content firmly in the former: to have a concept is to be able to make moves in a practice of giving and asking for reasons, and the moves are governed by norms of what one is entitled to say and what one is committed to by saying it. Brandom (1994) turned the metaphor into a theory. The content of a claim is its inferential role, the set of what it follows from and what follows from it, and that role is constituted by the practical attitudes of participants who keep score of one another's commitments. Language on this view is not a code for transmitting representations but a normative space in which positions are taken and assessed. The later Brandom (2008) added a piece of architecture that will matter below: the idea that for any vocabulary one can ask what practices or abilities are sufficient to deploy it, and can specify those abilities in a pragmatic metavocabulary that says what one must do to count as saying what the vocabulary says. Meaning, in that framework, is systematically related to use by relations that can be charted, and the charting is a kind of translation between what a vocabulary means and what one has to be able to do to mean it.

Now to the idea of conceptual engineering. The field has been given a canonical shape by Cappelen (2018) and by the collection edited by Burgess, Cappelen, and Plunkett (2020), and its recent state has been surveyed by Koch, Löhr, and Pinder (2023) and by Isaac, Koch, and Nefdt (2022). Three strands can be distinguished by their rationale for revision. The Carnapian strand takes revision to be explication: a vague or inconsistent concept is replaced by an exact one that is similar enough to the original, fruitful for theory, and as simple as fruitfulness allows (Carnap, 1950, ch. 1). The ameliorative strand, whose paradigm is Haslanger (2000), takes revision to be political: a concept is replaced by one that better serves the legitimate purposes of the people who use it, where the purposes are moral and social

¹ The distinction between e-representation, which is a matter of a token's covariation with an environmental condition, and i-representation, which is a matter of a token's inferential and functional place in a system, is drawn in Price (2013, ch. 2). The distinction is what allows Price to be a global expressivist without denying that some vocabularies track: tracking is a job, and the mistake of representationalism is to treat it as the job. The argument of Section 5 is that the two notions correspond to two classes in a type hierarchy, the first extending the second.

rather than theoretical. The externalist strand, which is Cappelen's own, takes revision to be metasemantic: the meaning of a term is fixed by factors that outrun any speaker's intentions, revision is the alteration of those factors, and the alteration is difficult in ways that the other strands underestimate. What is striking, once the three are laid side by side, is that the first two give a functional rationale for revision and the third gives none. Carnap's fruitfulness is a functional notion, since a concept is fruitful when it lets one state laws and draw inferences one could not state or draw before. Haslanger's amelioration is a functional notion, since a concept is improved when it does better the work we need it for. Cappelen's account tells us what revision consists in and why it is hard, and it is agnostic about why one would want it.²

The last few years have seen the functional rationale become a self-conscious programme. Thomasson (2020) proposes a pragmatic method for normative conceptual work that assesses concepts by the functions they serve and answers the charge of arbitrariness by pointing out that functions are not arbitrary. Nado (2021) argues for a radical functionalism on which even revisions that change the subject are permitted, provided that every function the old concept served is served somewhere in the new scheme. Simion and Kelp (2020) recast the whole enterprise as conceptual innovation assessed by etiological function. Queloz (2022) presses the authority problem, that is, the question of why anyone should defer to the concepts an engineer recommends, and argues that only an account of what the concept does for the people who use it can answer it. Löhr (2021) proposes that what engineers alter is not representations but linguistic commitments and entitlements, and Jorem and Löhr (2024) generalise the proposal into an inferentialist conceptual engineering whose objects are inferential rather than representational devices. A special issue of *Inquiry* has now been devoted to the relation between conceptual engineering and pragmatism (Henne; Huetter-Almerigi, 2024), and Löhr (2025) has asked directly what a pragmatist conceptual engineering would be.

I want to draw a conclusion from this list that its members mostly stop short of drawing. The functional turn is not a new methodological option within conceptual engineering. It is the recovery of the rationale that the practice always had and that the reference-fixing picture had obscured. Every case that anyone has offered as a success of engineering is a case in

² Chalmers (2025) gives a taxonomy on which engineering is distinguished from analysis by its aim of improvement rather than description, and he leaves open what the standard of improvement is. My claim is that once the standard is filled in, on any reading that has been proposed, it is functional. The externalist strand's silence on this point is not a defect of the strand so much as a sign that it is answering a different question, namely what changing a meaning consists in, and that question can be answered without saying why anyone would want to.

which a concept came to do a job better, and no case has ever been offered in which a concept came to refer more correctly, for the good reason that there is no such thing. That claim is Price's, and Brandom's, and before them Wittgenstein's, and it is the claim on which the very idea of a normative assessment of concepts depends. To the question of how far conceptual engineering rests on pragmatism, the answer is: as far as it rests on anything that would give one a reason to engineer. What the pragmatist inheritance does not supply, and what the functional turn has struggled to supply from its own resources, is a method for assessing revisions that is more than a list of desiderata. Thomasson is right that functions are not arbitrary. She has not said how to check that a proposed revision serves them. Nado is right that continuity of function is the constraint. She has not said how to tell that continuity has been preserved. The remainder of this article is an attempt to say.

3. Reference-Fixing and Its Discontents

The reference-fixing picture deserves a fair statement before it is set aside, and Cappelen's Austerity Framework is the fairest available. On that framework the meaning and reference of a term are determined by a vast and largely unknown set of factors, including the history of the term's use, the intentions of speakers now dead, the causal contacts between the community and its environment, and the deference relations among its members. No speaker controls these, no speaker knows more than a fragment of them, and so any attempt to change what a term means is an attempt to move an object whose mass one cannot measure by pushing on it with a force one cannot direct. Cappelen calls the first feature lack of control and the second inscrutability, and he concludes that conceptual engineering is nonetheless worth doing, in the way that trying to influence the climate is worth doing, without anyone being in a position to say what any particular intervention will bring about.³

The framework has drawn a series of responses, and it will be useful to see what they have in common. Koch (2021) formulates the externalist challenge precisely and argues that it can be met by distinguishing the semantic mechanisms externalism describes from the collective actions that can alter their inputs. Deutsch (2020) presses a dilemma: either engineers know how to change the semantic meaning of terms, in which case they owe an account they have

³ Cappelen (2018, part II) develops the framework and its two features. Inscrutability is the stronger of the two and the less often discussed: even a community that somehow controlled the factors would not, on the framework, be in a position to know what they had brought about. The translation of Section 4 has nothing to say about inscrutability so understood, because it locates the object of engineering somewhere the community can inspect, and it can inspect it because it put it there.

not given, or they merely stipulate new speaker meanings, in which case the achievement is trivial and philosophically inert. Pinder (2021) accepts the second horn and denies that it is inert, arguing that conceptual engineering operates at the level of speaker meaning and that this makes it easier, not emptier. Sawyer (2020) separates concepts from words, holds that concepts are individuated by the worldly kinds they track, and concludes that what engineers change is which concept a word expresses rather than any concept itself. Nado (2023) proposes to take control by doing conceptual engineering without much metasemantics at all. Riggs (2019) tells engineers to stop worrying about externalism. The responses disagree about almost everything, and they agree about one thing so completely that the agreement is invisible. Each takes the object at issue to be a relation between a term and a portion of the world, and each asks how, whether, and at what level that relation can be moved.

The pragmatist does not deny that there are such relations. The pragmatist denies that they are what the engineer alters, and the denial is best made on a case. In August 2006 the International Astronomical Union adopted a resolution defining a planet as a body that orbits the Sun, has enough mass to have assumed a nearly round shape, and has cleared the neighbourhood around its orbit, and it introduced the further category of dwarf planet for bodies that satisfy the first two conditions but not the third (International Astronomical Union, 2006). Pluto fell into the second category. Described in the reference-fixing idiom, the IAU discovered, or decided, that planet had eight things in its extension rather than nine. But that description misses what happened. What the resolution altered was what one is entitled to infer from the claim that something is a planet and what one is committed to by making the claim. Before 2006, calling a body a planet committed one to nothing about its dynamical relation to its neighbours. After 2006, it committed one to the claim that the body is gravitationally dominant in its orbital zone, and it entitled one to infer that the body formed by a process that cleared that zone. The extension changed as a consequence of the change in commitments, and it changed in the way a shadow changes when the object casting it is moved. Nobody on the floor in Prague was trying to alter a shadow.

This is the point at which the pragmatist slogan that there is always more than one way to go on earns its keep. The slogan descends from Wittgenstein's remarks on following a rule, and in particular from the observation at *Philosophical Investigations* 201 that no course of action can be determined by a rule, because every course of action can be made to accord with it (Wittgenstein, 1953). I do not want to enter the exegetical argument about what Wittgenstein concluded from this, and I do not need to, because Brandom (1994, ch. 1) has already drawn

the moral that matters here. The regress of interpretations shows that a norm cannot be a further rule, and the repair is to locate the norm in the practical attitudes of those who treat performances as correct and incorrect. On that repair, a concept is not a rule that determines its own application to every new case. It is a standing that a community confers, and the community could, without any change in the world, confer it differently. The IAU could have kept nine planets by treating orbital clearance as irrelevant. It could have admitted several dozen by dropping the roundness condition. Each of these would have been a way of going on with the word planet, and none of them would have been the correct one, because correctness in this sense is not a property that ways of going on have prior to a community's choosing among them. What ways of going on do have, prior to the choice, is a set of jobs they would do well or badly, and it is by those that a choice is assessed.⁴

Once the object of engineering is seen to be a structure of commitments rather than a word-world relation, the responses to Cappelen rearrange themselves. Lack of control is the observation that a community does not control the world. That is true and irrelevant, since the community controls what it commits itself to when it applies a term, which is what it was trying to control. Deutsch's dilemma between semantic meaning and speaker meaning becomes a distinction between the commitment structure in force in a community and the one a speaker proposes, and the gap between them is the implementation problem (Jorem, 2021), which is a problem about persuasion rather than about metasemantics. Sawyer's separation of concepts from words survives, but her identification of concepts with the kinds they track becomes one option among several: some concepts have tracking as their job, and for those the identification is apt, and others do not. What does not survive is the idea that the engineer's success is the arrival of a reference relation at a destination. There is no destination, and there are jobs.

4. From Natural-to-Artificial to Normative-to-Functional

4.1 The precedent

⁴ I am taking Brandom's normative pragmatics as the background rather than defending it, in keeping with the division of labour of this article. Readers who prefer a different resolution of the rule-following considerations will find that the argument needs only the weak claim that the application of a concept to new cases is not fixed by the concept alone, and that claim is common to nearly every reading of the passage.

The pragmatist owes an account of assessment, and I propose to discharge the debt by imitating a procedure with a long history in the philosophy of language. Carnap's explication took a concept from natural language, where its application was governed by habits that nobody had stated, and rendered it in an artificial language, where its application was governed by rules that anybody could check. Montague's programme extended the procedure from concepts to constructions, translating fragments of English into a typed lambda calculus in which composition, scope, and entailment could be computed rather than intuited. In both cases the translation was not a discovery about the natural language. It was a proposal for a substitute that made explicit what the original had left implicit, and its warrant was that the substitute did a job the original could not. The same procedure, applied not to the semantics of a concept but to its normative role, yields what I will call the normative-to-functional translation.

The immediate model is a piece of my own work, in which the two-dimensional framework of Chalmers (2006) and Jackson (1998) was rendered in Haskell and the rendering used to argue for the superiority of a typed over an untyped functional host (Alonso, 2026a). The argument there did not turn on elegance. It turned on the observation that the framework depends on keeping two roles of a possible world apart, the world considered as actual and the world considered as counterfactual, and that a language with a type system can enforce the distinction by wrapping one underlying type in two others that the compiler refuses to confuse, whereas a language without one can only recommend the distinction and hope the programmer observes it. The lesson generalises. Whatever a formal semantics is for, the thing a strongly typed language adds is the difference between a discipline that is stated and a discipline that is enforced. The normative-to-functional translation exploits that difference at the level where conceptual engineering operates, which is the level of the roles concepts play.

4.2 The interface and its instances

Haskell has a mechanism, the type class, that was introduced by Wadler and Blott (1989) to make ad hoc polymorphism principled and that has since become the language's primary device for abstraction. A type class declares an interface: a set of operations that any type

belonging to the class must supply, together with laws, stated in comments or checked by property-based testing, that the operations must satisfy. A type joins the class by an instance declaration that supplies the operations. Code written against the class can be run over any instance and cannot inspect anything about a type that the class does not expose. The translation I propose reads a concept as a class and a way of applying it as an instance. Here is the interface in the least decorated form I can give it.

```
class Kind k where
  falls    :: k -> Case -> Bool
  commits   :: k -> Case -> [Claim]
  entitles  :: k -> Case -> [Claim]

  -- Law 1: commits k c is non-empty only if falls k c.
  -- Law 2: entitles k c is non-empty only if falls k c.
  -- Law 3: no claim in commits k c is the negation
  --        of a claim in entitles k c.
```

A Kind is anything for which one can say whether a case falls under it, what one is committed to by saying that it does, and what one is entitled to infer. The three laws are the minimal normative content of any concept whatever: one is not committed by a term one has not applied, one is not entitled by it either, and a concept that commits its user to a claim while entitling the same user to its denial is inconsistent in Scharp's (2013) sense. Everything specific to a particular concept goes into its instances, and the planet case gives two.

```
data Planet2005 = Planet2005
data Planet2006 = Planet2006

instance Kind Planet2005 where
  falls    _ c = orbitsSun c && nearlyRound c
  commits   _ c = [Claim "orbits the Sun" | falls Planet2005 c]
  entitles  _ c = [Claim "in hydrostatic equilibrium"
                  | falls Planet2005 c]

instance Kind Planet2006 where
  falls    _ c = orbitsSun c && nearlyRound c
                  && clearedNeighbourhood c
  commits   _ c = [Claim "dynamically dominant in its zone"
                  | falls Planet2006 c]
  entitles  _ c = [Claim "formed by accretion that cleared its zone"
                  | falls Planet2006 c]
```

Three features of this rendering do argumentative work. The first is that the interface is unchanged across the revision. Any function elsewhere in the program whose type is `Kind k => k -> ...` was written against the interface and will accept Planet2006 wherever it accepted

Planet2005. That is the formal residue of continuity of subject. Strawson (1963, p. 506) objected to Carnap that explication changes the subject, and the functional turn has replied that the subject is preserved when the function is (Nado, 2021, sec. 3). The type system makes the reply accurate. The subject is the interface, and the compiler will tell you, by refusing to build, whether a revision has changed it. The second feature is that the two instances are both admissible, in the sense that both satisfy the laws, and that neither is correct, in the sense that nothing in the interface prefers one. This is the pragmatist's plurality of ways of going on rendered as a property of the program rather than as an interpretive claim. The third feature is that the instances differ in what they commit and entitle their users to, and it is in that difference, and not in any difference of extension, that the revision consists. The extension is recoverable, as the set of cases for which falls returns True, but it is derived from the instance and is not what the instance is.

Assessment now has a form. A proposed revision is a new instance, and it can fail in exactly two ways. It can fail to satisfy the laws of the interface, in which case it is not a concept at all, and a property-based test in the style of Claessen and Hughes (2000) will find the counterexample. Or it can satisfy the laws and fail to serve the purposes for which the concept was wanted, in which case the failure shows up where those purposes are coded: in the functions written against the interface that produce the wrong result when run over the new instance. The IAU's instance passes the laws, and it passes the second test for the community of dynamicists who wanted a category that tracked orbital dominance, and it fails the second test for the community of planetary geologists who wanted a category that tracked internal differentiation and would have been happy to keep Pluto. There was, in Prague, more than one way to go on, and the choice among them was a choice about whose functions would be served. The translation does not make that choice. It makes it visible, and it makes the consequences of each option computable in advance rather than discoverable in retrospect.⁵

The ameliorative case renders the same way, and the rendering is instructive about a standing complaint against it. Haslanger's (2000) proposal was that to be a woman is to occupy a position of systematic subordination marked by presumed female sex, and the complaint,

⁵ The two communities are not my invention. The dynamical criterion was championed by those who study orbital evolution, and the objection from planetary geology, that the criterion demotes a differentiated, geologically active body on the basis of its neighbours rather than its own properties, was made at the time and has been repeated since. The point for present purposes is that both parties accepted the interface and disagreed about the instance, which is the situation the translation is built to describe.

made in various registers, is that this changes the subject: whatever the proposed concept is, it is not what anyone meant by the word. In the translation, the proposal is a new instance under an unchanged interface. The operation falls now consults social position where it once consulted biology, commits now returns claims about subordination where it once returned claims about reproductive role, and entitles licenses inferences about how a case is treated where it once licensed inferences about how a case is constituted. Every function in feminist theory written against the interface, every argument that took a case's being a woman as a premise and drew a conclusion from what that commits and entitles one to, continues to build. The complaint that the subject has changed is the complaint that the instance has changed, and the instance is exactly what an amelioration was supposed to change. Whether the new instance is a good one is a question about the jobs the concept was wanted for, and Haslanger's own defence is, in these terms, an argument that the tracking of reproductive biology was never among the jobs that mattered to the theory she was doing, so that an instance which drops it loses nothing the theory needed. One may disagree with that assessment. What one cannot do, once the interface is in view, is claim that the disagreement is about whether the word still refers, because both instances are ways of going on with the word, and neither of them refers in any sense the other does not.

4.3 Dispatch, negotiation, and Julia

Haskell resolves which instance is in force at compile time, from the declared type of the expression. That is a good model of a stipulated context: a community that has declared, as the IAU did, which concept it is using. It is a poor model of the more common situation in which the concept in force is not settled in advance and is being negotiated in the exchange itself. Plunkett and Sundell (2013) have argued that a good deal of apparent disagreement about matters of fact is metalinguistic negotiation about which concept a word should express, and that such negotiation is a legitimate way of settling the concept rather than a confusion to be cleared up. A language that models negotiation needs the instance to be chosen late, from what actually turns up, rather than early, from what was declared.

Julia offers such a language. Its central design decision, described by Bezanson, Edelman, Karpinski, and Shah (2017), is multiple dispatch: a function is a name for a family of methods, and the method that runs is selected at the call from the runtime types of all its arguments together. Rendered in Julia, the planet concept is a generic function with a method per way of going on.

```

abstract type Kind end
struct Planet2005 <: Kind end
struct Planet2006 <: Kind end

falls(::Planet2005, c) = orbitssun(c) && nearlyround(c)
falls(::Planet2006, c) = orbitssun(c) && nearlyround(c) &&
                        clearedneighbourhood(c)

commits(::Planet2005, c) = falls(Planet2005(), c) ?
    ["orbits the Sun"] : String[]
commits(::Planet2006, c) = falls(Planet2006(), c) ?
    ["dynamically dominant in its zone"] : String[]

```

The philosophical payoff is in what happens when dispatch fails. If two methods apply equally well to a call and neither is more specific than the other, Julia raises a method ambiguity error and refuses to choose. That refusal is the formal counterpart of a metalinguistic negotiation that has not been concluded: the parties have each supplied a way of going on, the case in front of them is one on which the ways diverge, and nothing in the language settles which is in force. A community whose concept of planet contained both instances, with no rule of precedence between them, would have exactly this status with respect to Pluto. The resolution of 2006 can be read as the addition of a precedence rule, and the years of dispute before it as a standing ambiguity error that astronomers had learned to live with. What Haskell models as a declaration, Julia models as an outcome, and the two together cover the cases the engineer meets.⁶

4.4 *Instances as proofs*

There is a further reading of the translation that ties it to the Curry-Howard correspondence and that clarifies what an engineered concept computes. Under the correspondence, in the form Howard (1980) gave it and Wadler (2015) has expounded, a type is a proposition and a term of that type is a proof of it. A class declaration with laws is then a proposition of a particular kind: it asserts that there is a way of applying the concept that meets the stated obligations. An instance declaration is a proof of that proposition, a constructive demonstration that the obligations can be met, produced by supplying operations that meet

⁶ The choice between Haskell and Julia here is not the choice made in Alonso (2026a) between Haskell and Scheme, and the reasons differ. There the question was whether a distinction could be enforced by types or only by convention, and Haskell won. Here the question is whether the instance in force should be settled by declaration or by the encounter, and the two languages model the two answers. A Lisp dialect would model a third possibility, in which the concept itself is data and can be rewritten by the program that runs it, and that would be a model of a community that engineers its concepts on the fly. I set it aside because the cases at issue in the engineering literature are not, on the whole, of that kind.

them. Two instances are two proofs of the same proposition. The pragmatist's slogan now has a precise form. To say that there is always more than one way to go on is to say that the proposition an interface expresses is not proof-irrelevant: it matters, for the purposes the concept serves, which proof of it one holds, and distinct proofs have distinct consequences downstream even though they establish the same claim.

I have argued elsewhere that the Curry-Howard reading of a modal principle reveals what that principle computes, and that the Barcan formulas, read as types, describe transformations on proofs rather than facts about the population of worlds (Alonso, 2026b). The present case is parallel. What an engineered concept computes, read as a type, is a transformation on commitments: it takes a case and returns the claims one has undertaken and the claims one may draw. A revision is a change in the transformation with the type held fixed. And the type has consequences of its own, independent of any instance. Wadler's (1989) theorems for free, which follow from Reynolds's (1983) parametricity, say that from the type of a polymorphic function alone one can derive theorems that hold of it under every instantiation. Applied here, the result is that any function written against `Kind k => k` provably cannot depend on any feature of the concept not exposed by the interface. That is the theorem behind the reply to Strawson. Whatever was said using the old concept that made use only of the interface can be said, with the same warrant, using the new one, and this is not a hope but a consequence of the type.

5. Do We Need Reference?

The question in my title can now be answered, and the answer is neither yes nor no but a specification of when. Consider a second class, which extends the first.

```
class Kind k => Tracks k where
  extension :: k -> World -> Set Case

  -- Law 4: c is in extension k w exactly when
  --       falls k c holds at w.
```

A `Kind` that is also an instance of `Tracks` has, among its operations, one that returns the set of cases falling under it at a world. That operation is a reference relation in the only sense the reference-fixing picture ever needed: a function from the concept and a world to a portion of the world. Several things follow from placing it here rather than at the foundation.

The first is that reference is a function some concepts perform, in Price's sense of function, and not a relation all concepts stand in. Planet2006 is an instance of Tracks, because the community that adopted it wanted a category whose extension could be read off the dynamical facts, and the whole point of the roundness and clearance conditions was to make the extension a matter of measurement. Price's e-representational vocabularies are the ones whose interface includes Tracks. His i-representational vocabularies, the ones whose job is to express a stance or to license inferences rather than to covary with an environmental condition, are Kinds that are not Tracks. The concept of a promise, or of a probability, or of a possibility, commits and entitles its user without there being an extension anyone was trying to measure, and the translation lets one say so by declining to write the instance. This is functional pluralism made into a hierarchy of classes, and it is, I think, what Price was after when he insisted that the representational idiom be earned by a vocabulary rather than assumed of it.⁷

The second consequence is that even for concepts that track, reference is instance-level and derived. Planet2006 has an extension because it is an instance of Tracks, and the extension it has is fixed by its falls operation together with the world. The two-dimensional blueprint of my earlier work makes the same point about names: the primary and secondary intensions of Hesperus are not two relations the name stands in but two slices of one function, obtained by holding a different argument fixed (Alonso, 2026a). Reference, on both renderings, is what you get when you evaluate a concept at a world, and the concept is prior to the evaluation. The parallel can be pressed a step further, because the two-dimensional framework turns out to have anticipated the structure of the translation. The primary intension of a term, on the epistemic reading, is what a competent speaker grasps a priori: the condition a case must meet to fall under the term, considered as a condition on however the world turns out to be. That is the interface, or as much of the interface as concerns falls. The secondary intension is what the term picks out across counterfactual worlds once the actual world has been fixed, and that is the extension operation of a particular instance evaluated with the actual world as argument. Two-dimensionalism was built to explain how a term could have a single a priori role and a posteriori reference, and the explanation was

⁷ Löhr (2021) describes his account as non-representationalist but not anti-representationalist, and the hierarchy of Section 5 is a way of making that description precise: the Kind class is non-representational in that nothing in it mentions extension, and it is not anti-representational because Tracks extends it for those concepts that need it. The same hierarchy answers a worry Jorem and Löhr (2024) raise about their own proposal, that an inferentialist engineering has no place for the intuition that some concepts answer to the world. It has a place, and the place is a subclass.

that the role is a function and the reference is its value at an argument nobody chooses. Conceptual engineering, read through the translation, is the practice of altering the function while leaving the argument to the world, and the reason it has seemed puzzling that engineers could alter reference is that they cannot, and were never trying to. It follows that the engineer never engineers reference. The engineer engineers the instance, and the world supplies the reference by supplying the argument. That is why lack of control was never a problem. Nobody wanted control over the argument, and the community has complete control over the function.

The third consequence answers the question. Conceptual engineering needs the metasemantic category of reference for a concept exactly when the concept's interface includes Tracks, that is, when tracking a worldly condition is among the jobs the concept was wanted for. For such concepts a revision is assessable partly by whether the new extension function serves the tracking purpose, and the literature's attention to natural-kind terms, to knowledge, and to planet is well placed. For concepts that do not track, the category is idle, and the attempt to assess a revision by whether it gets the reference right is an attempt to run extension on a type that has no instance of it. The type checker will report the error, and so, less reliably, will the philosophical literature, which has spent a good deal of energy asking whether ameliorated normative concepts refer correctly without first asking whether reference was ever their job. Sawyer's (2020) view that concepts are individuated by the kinds they track is, in the vocabulary of the translation, the view that every Kind is a Tracks. It is a substantive claim, and it is false of the vocabularies Price spent his career on.

6. Objections

The first objection is that the translation is functionalism restated in code and adds nothing to what Thomasson, Nado, and Löhr have already said. The reply is that the addition is enforcement. Thomasson says that concepts are assessed by their functions, and the translation says which functions, in the form of an interface with laws that a revision can be tested against. Nado says that continuity of function is the constraint, and the translation says what continuity is, in the form of a theorem that any consumer of the interface is invariant across instances. Löhr says that engineers alter commitments and entitlements, and the translation says how, in the form of two operations whose values a revision changes and a third whose value it may leave alone. Each of these is a claim that was asserted in prose

and is checked in the typed rendering. The difference between asserting a discipline and enforcing it is the difference the blueprint of Alonso (2026a) was built to exhibit.

The second objection is that type classes are closed and concepts are open-textured. Waismann's (1945) point that no empirical concept is defined for every possible case is well taken, and a class with a fixed interface looks like a denial of it. But type classes are open in the respect that matters: instances can be added to a class at any point in a program, by anyone with access to the interface, without the class's author having anticipated them. What is fixed is the interface, and the interface is fixed only in the sense that a revision of it is a different and more radical kind of engineering, the replacement of what a concept is for rather than of how it does it. The translation handles that too, by superclass constraints and by deprecation, but it handles it as a distinct operation, and I count the distinction a virtue. The functional turn has tended to run together revisions that keep a concept's job and change its method with revisions that change its job, and the type system will not let one do that.⁸

The third objection is that without reference there is no truth, and without truth there is no assessment. The reply comes in two parts. For concepts that track, reference is retained by the Tracks class and truth is retained with it, since a claim that a case falls under such a concept is true at a world exactly when the case is in the extension there. For concepts that do not track, the pragmatist has never needed a robust notion of truth to assess them, because their assessment was always by what they let one do, and Price's (2011) deflationism about truth is the standing account of how one can say that a normative claim is true without thereby claiming that it tracks. The objection assumes that assessment must be assessment for truth, which is the representationalist assumption the whole of Section 2 was written to dislodge.

The fourth objection is reflexive: the translation is itself an engineering of the concept of a concept, and so it presupposes the practice it was meant to underwrite. I accept the description and deny that it is a difficulty. The translation is an explication, and explications are proposals that make their case by their fruits. If reading concepts as interfaces lets one say precisely what continuity of subject is, what a plurality of ways of going on comes to, and

⁸ The same discipline blocks a move the ameliorative literature has sometimes been tempted by, in which a concept is kept in name while its interface is replaced wholesale, so that the new concept commits and entitles its users to nothing the old one did. That is not a revision of a concept but the introduction of a homonym, and the translation will not type it as the former. Whether the introduction of a homonym can be a good idea is a separate question, and often the answer is yes, but it is a different question from the one the functional turn was asking.

when reference is needed, then the reading has earned its place in the way any concept earns one, and the pragmatist, of all people, is not entitled to ask for more.

7. Conclusion

I have argued that conceptual engineering rests on pragmatist views of language to the full extent that it rests on anything that would give one a reason to engineer, and that the reference-fixing picture, whatever its merits as metasemantics, describes a practice nobody is engaged in. The engineer alters what a community commits itself to and entitles itself to when it applies a term, and the alteration is assessed by the jobs the term was wanted for. That leaves the pragmatist with the task of saying how such an assessment could be anything but a matter of taste, and the normative-to-functional translation is my answer. Read a concept as an interface with laws, a way of applying it as an instance, and a revision as a change of instance under the interface, and the pragmatist's claims about plurality, continuity, and function become properties that a typed language checks. There is more than one way to go on because an interface admits more than one instance. The subject is preserved because the interface is, and the compiler will say so. Reference is needed when, and only when, the interface includes it.

The translation does not choose among ways of going on. Nothing formal could, since the choice is a choice about whose purposes will be served, and that is a matter for the people whose purposes they are. What the translation offers is the thing that Carnap offered the logicians and Montague offered the linguists: a substitute exact enough to be checked, in a place where the original could only be argued about. The pragmatist has always said that language is a tool. It is a mild irony, and I hope a productive one, that the tool best suited to saying what that means turns out to be a language.

Bibliographic References

ALONSO, B. Formalising two-dimensional semantics: a Haskell blueprint. 2026a. Forthcoming in *Kriterion*.

ALONSO, B. Necessitism as a Structural Rule: Persistence and the Barcan Formulas under Curry—Howard. 2026b. Forthcoming in *Journal of Philosophical Logic*.

BEZANSON, Jeff; EDELMAN, Alan; KARPINSKI, Stefan; SHAH, Viral B. Julia: a fresh approach to numerical computing. *SIAM Review*, Philadelphia, v. 59, n. 1, p. 65-98, 2017. DOI: 10.1137/141000671.

BRANDOM, Robert B. *Making it explicit: reasoning, representing, and discursive commitment*. Cambridge, MA: Harvard University Press, 1994.

BRANDOM, Robert B. *Between saying and doing: towards an analytic pragmatism*. Oxford: Oxford University Press, 2008.

BURGESS, Alexis; CAPPELEN, Herman; PLUNKETT, David (Org.). *Conceptual engineering and conceptual ethics*. Oxford: Oxford University Press, 2020.

CAPPELEN, Herman. *Fixing language: an essay on conceptual engineering*. Oxford: Oxford University Press, 2018.

CARNAP, Rudolf. *Logical foundations of probability*. Chicago: University of Chicago Press, 1950.

CHALMERS, David J. The foundations of two-dimensional semantics. In: GARCÍA-CARPINTERO, Manuel; MACIÀ, Josep (Org.). *Two-dimensional semantics*. Oxford: Oxford University Press, 2006. p. 55-140.

CHALMERS, David J. What is conceptual engineering and what should it be? *Inquiry*, Abingdon, v. 68, n. 9, p. 2902-2919, 2025. DOI: 10.1080/0020174X.2020.1817141.

CLAESSEN, Koen; HUGHES, John. QuickCheck: a lightweight tool for random testing of Haskell programs. In: ACM SIGPLAN INTERNATIONAL CONFERENCE ON FUNCTIONAL PROGRAMMING, 5., 2000, Montreal. *Proceedings...* New York: Association for Computing Machinery, 2000. p. 268-279. DOI: 10.1145/351240.351266.

DEUTSCH, Max. Speaker's reference, stipulation, and a dilemma for conceptual engineers. *Philosophical Studies*, Dordrecht, v. 177, n. 12, p. 3935-3957, 2020. DOI: 10.1007/s11098-020-01416-z.

HASLANGER, Sally. Gender and race: (what) are they? (What) do we want them to be? *Noûs*, Boston, v. 34, n. 1, p. 31-55, 2000. DOI: 10.1111/0029-4624.00201.

HENNE, Céline; HUETTER-ALMERIGI, Yvonne. Conceptual engineering and pragmatism: historical and theoretical perspectives. *Inquiry*, Abingdon, v. 67, n. 3, p. 811-823, 2024.

HOWARD, William A. The formulae-as-types notion of construction. In: SELDIN, Jonathan P.; HINDLEY, J. Roger (Org.). *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*. London: Academic Press, 1980. p. 479-490.

INTERNATIONAL ASTRONOMICAL UNION. *Resolution B5: definition of a planet in the Solar System*. Prague: IAU General Assembly, 2006.

ISAAC, Manuel Gustavo; KOCH, Steffen; NEFDT, Ryan. Conceptual engineering: a road map to practice. *Philosophy Compass*, Hoboken, v. 17, n. 10, e12879, 2022. DOI: 10.1111/phc3.12879.

JACKSON, Frank. *From metaphysics to ethics: a defence of conceptual analysis*. Oxford: Oxford University Press, 1998.

- JOREM, Sigurd. Conceptual engineering and the implementation problem. *Inquiry*, Abingdon, v. 64, n. 1-2, p. 186-211, 2021. DOI: 10.1080/0020174X.2020.1809514.
- JOREM, Sigurd; LÖHR, Guido. Inferentialist conceptual engineering. *Inquiry*, Abingdon, v. 67, n. 3, p. 932-953, 2024. DOI: 10.1080/0020174X.2022.2062045.
- KOCH, Steffen. The externalist challenge to conceptual engineering. *Synthese*, Dordrecht, v. 198, n. 1, p. 327-348, 2021.
- KOCH, Steffen; LÖHR, Guido; PINDER, Mark. Recent work in the theory of conceptual engineering. *Analysis*, Oxford, v. 83, n. 3, p. 589-603, 2023.
- LÖHR, Guido. Commitment engineering: conceptual engineering without representations. *Synthese*, Dordrecht, v. 199, n. 5-6, p. 13035-13052, 2021. DOI: 10.1007/s11229-021-03365-4.
- LÖHR, Guido. What is pragmatist conceptual engineering? *Inquiry*, Abingdon, 2025. Advance online publication. DOI: 10.1080/0020174X.2025.2495643.
- MONTAGUE, Richard. The proper treatment of quantification in ordinary English. In: HINTIKKA, Jaakko; MORAVCSIK, Julius; SUPPES, Patrick (Org.). *Approaches to natural language*. Dordrecht: Reidel, 1973. p. 221-242.
- NADO, Jennifer. Conceptual engineering, truth, and efficacy. *Synthese*, Dordrecht, v. 198, suppl. 7, p. 1507-1527, 2021. DOI: 10.1007/s11229-019-02096-x.
- NADO, Jennifer. Taking control: conceptual engineering without (much) metasemantics. *Inquiry*, Abingdon, v. 66, n. 10, p. 1974-2000, 2023. DOI: 10.1080/0020174X.2020.1850342.
- PINDER, Mark. Conceptual engineering, metasemantic externalism and speaker-meaning. *Mind*, Oxford, v. 130, n. 517, p. 141-163, 2021. DOI: 10.1093/mind/fzz069.
- PLUNKETT, David; SUNDELL, Tim. Disagreement and the semantics of normative and evaluative terms. *Philosophers' Imprint*, Ann Arbor, v. 13, n. 23, p. 1-37, 2013.
- PRICE, Huw. *Naturalism without mirrors*. Oxford: Oxford University Press, 2011.
- PRICE, Huw. *Expressivism, pragmatism and representationalism*. With Simon Blackburn, Robert Brandom, Paul Horwich and Michael Williams. Cambridge: Cambridge University Press, 2013.
- QUELOZ, Matthieu. Function-based conceptual engineering and the authority problem. *Mind*, Oxford, v. 131, n. 524, p. 1247-1278, 2022. DOI: 10.1093/mind/fzac028.
- REYNOLDS, John C. Types, abstraction and parametric polymorphism. In: MASON, R. E. A. (Org.). *Information processing 83*. Amsterdam: North-Holland, 1983. p. 513-523.
- RIGGS, Jared. Conceptual engineers shouldn't worry about semantic externalism. *Inquiry*, Abingdon, 2019. Advance online publication. DOI: 10.1080/0020174X.2019.1675534.
- SAWYER, Sarah. The role of concepts in fixing language. *Canadian Journal of Philosophy*, Cambridge, v. 50, n. 5, p. 555-565, 2020.

SCHARP, Kevin. *Replacing truth*. Oxford: Oxford University Press, 2013.

SELLARS, Wilfrid. Some reflections on language games. *Philosophy of Science*, Chicago, v. 21, n. 3, p. 204-228, 1954. DOI: 10.1086/287344.

SELLARS, Wilfrid. Empiricism and the philosophy of mind. In: FEIGL, Herbert; SCRIVEN, Michael (Org.). *Minnesota studies in the philosophy of science*. Minneapolis: University of Minnesota Press, 1956. v. 1, p. 253-329.

SIMION, Mona; KELP, Christoph. Conceptual innovation, function first. *Noûs*, Boston, v. 54, n. 4, p. 985-1002, 2020. DOI: 10.1111/nous.12302.

STRAWSON, Peter F. Carnap's views on constructed systems versus natural languages in analytic philosophy. In: SCHILPP, Paul Arthur (Org.). *The philosophy of Rudolf Carnap*. La Salle: Open Court, 1963. p. 503-518.

THOMASSON, Amie L. A pragmatic method for normative conceptual work. In: BURGESS, Alexis; CAPPELEN, Herman; PLUNKETT, David (Org.). *Conceptual engineering and conceptual ethics*. Oxford: Oxford University Press, 2020. p. 435-458. DOI: 10.1093/oso/9780198801856.003.0021.

WADLER, Philip. Theorems for free! In: INTERNATIONAL CONFERENCE ON FUNCTIONAL PROGRAMMING LANGUAGES AND COMPUTER ARCHITECTURE, 4., 1989, London. *Proceedings...* New York: Association for Computing Machinery, 1989. p. 347-359. DOI: 10.1145/99370.99404.

WADLER, Philip. Propositions as types. *Communications of the ACM*, New York, v. 58, n. 12, p. 75-84, 2015. DOI: 10.1145/2699407.

WADLER, Philip; BLOTT, Stephen. How to make ad-hoc polymorphism less ad hoc. In: ACM SIGPLAN-SIGACT SYMPOSIUM ON PRINCIPLES OF PROGRAMMING LANGUAGES, 16., 1989, Austin. *Proceedings...* New York: Association for Computing Machinery, 1989. p. 60-76. DOI: 10.1145/75277.75283.

WAISMANN, Friedrich. Verifiability. *Proceedings of the Aristotelian Society, Supplementary Volume*, London, v. 19, p. 119-150, 1945.

WITTGENSTEIN, Ludwig. *Philosophical investigations*. Translated by G. E. M. Anscombe. Oxford: Blackwell, 1953.

SUBMISSION METADATA

AUTHORSHIP CONTRIBUTION

The author assumes responsibility for all aspects of this article, in accordance with the [CRediT](#) taxonomy, from conceptualization, formal analysis, code snippets preparation, to finally drafting and writing the manuscript.

ACKNOWLEDGMENTS

The authors thank the reviewers and editors of *Sofia*.

FUNDING

This work did not receive funding from any funding agency.

USE OF ARTIFICIAL INTELLIGENCE

The authors declared that they did not use Generative Artificial Intelligence tools in the preparation of this manuscript.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest related to the publication of this article.

RESEARCH DATA AND OTHER MATERIALS AVAILABILITY

Data availability: The contents underlying the research text are fully contained in the manuscript.

LICENSE OF USE

Copyright (c) 2026 Bernardo Alonso – CC BY Attribution 4.0 International License. This is an open-access article distributed under the terms of the Creative Commons Attribution License

- **Texto completo:** [CC BY 4.0](#). Free use with the obligation to acknowledge authorship and original publication in this journal.
- **Metadados:** [CC0 1.0 Universal](#) (Public Domain) – free use, including for commercial purposes.

PUBLISHER

Universidade Federal do Espírito Santo. As ideias expressadas neste artigo são de responsabilidade das pessoas autoras, não representando, necessariamente, a opinião dos editores ou da universidade.
<https://periodicos.ufes.br/index/index>

EDITORS

EDITOR-IN-CHIEF: Jorge Luiz Viesenteiner [0000-0003-3727-7890](#)

Editing: Wellington Alan Soares Marques [0000-0002-6537-7680](#)

Bernardo Alonso

Bernardo Alonso is a Ph.D. from UFRJ, and full professor at UFMT. Worked as research assistant at University of Hertfordshire (UK) under the supervision of professor Luciano Floridi (Yale), worked at Phenomenal Qualities project (Oxford, Herts and Cambridge) with professor Gallen Strawson, Tim Crane and David Chalmers, is affiliated to Privacy International (UK), Electronic Frontier Foundation (USA) and Computer Chaos Camp (Germany).

The texts of this article have been reviewed by third parties and submitted for validation by the author(s) before publication