Logics of Metaphysical Definition

Andrew Bacon

First draft June 2023, updated October 15, 2025

Contents

1 Introduction

Philosophy often finds itself in the business of explaining one sort of entity — holes, numbers, knowledge, mental properties, or what have you — in terms of others considered to be more basic. Presupposed in this endeavor is the idea that reality has a hierarchical structure. Perhaps atoms are more basic than tables and chairs, material objects more basic than the shadows they cast, physical properties more basic than mental, the concrete world more basic than the platonic, and so on. Perhaps it is also possible to reach a point where no further reductions can be applied, in which case we reach the *fundamental* objects, properties and relations: entities that cannot be reduced to anything more basic.

One much-discussed account of the hierarchical structure of reality posits and theorizes in terms of a notion of *ground*: a relation between facts or propositions representing a form of metaphysical explanation — the fact that p is true metaphysically explains why q is true. In this paper I will outline and formalize a different sort of approach to metaphysical structure based instead on the notion of *metaphysical analysis*, found, for instance, in Russell.[1] Some version of this idea is implicit in much of the history of philosophy. The idea is that the less basic things can be metaphysically *defined*, or *analyzed*, in terms of the less basic things. Thus, if *knowledge* can defined from *justification*, *truth* and *belief* the latter are more basic than the former in the relevant metaphysical sense, if the property of *there being a hole in* can be defined from *being perforated* then the latter is more basic, and so on. While the logic of grounding is reasonably well-understood—see the classic treatment in Fine (2012))—less work has been done on the logic of this alternative conception of the hierarchical structure of reality, and I hope to fill this lacuna.

In Fine (2012) grounding is treated as a propositional connective representing a relation between propositions. However, since *knowledge*, *being in pain*, *the number 7* are expressed using expressions from different grammatical categories, or "types"—an operator, predicate and singular term respectively—and since the definiens and definiendum of a metaphysical analysis can belong to differing categories, we need something more general than a connective to express this relation. We must conduct our investigation within the framework of type theory. According to type theories, expressions

[1] See Russell (1940) §1.

have types which impose constraints on which expressions of the language it can be combined with using the syncategorematic operations of the language. A common form of type theory is functional type theory, that applies to languages that at least have application of a predicate (or other functional expression) to an argument as a syncategorematic mode of combination. The functional types include types e and t, corresponding to the type of names and sentences respectively, and given any two types σ and τ, there is a functional type $\sigma \to \tau$: the type of expressions that when combined with an expression of type σ via application produces an expression of type τ.

The objective of this paper is to propose a general and neutral framework for thinking about propositional structure in which an account of metaphysical definability can be given. Much attention has been paid, recently, to the fact that certain very fine-grained accounts of propositional structure are known to be inconsistent.[2] These results, however, do not show that all structural notions are necessarily incoherent. The present approach extends the project in Bacon (2019) of taking a notion that *would* be well-defined if propositions, properties and relations were very fine grained, and instead taking that notion as primitive and theorizing in terms of its abstract properties, allowing us to apply them contexts where less structure is being assumed. The notion we take as our starting point is the notion of a *logical form*—itself an instance of the more general notion of a propositional function. We represent these by augmenting a model of type theory with a distinguished class of functions between the domains representing the logical forms. On a structured picture a propositional function could be thought of as a structured proposition some of whose constituents have been punched out leaving "holes" where entities can be inserted — a function whose outputs range over entities with a common structure parametrized by the entities filling certain gaps — and a logical form a propositional function in which all constituents have been punched out. While talk of logical forms, gaps and common structure seem to assume a very structured picture, one can take the notion of a logical form as basic to be closed under certain general operations like composition. A given proposition need not in general have a unique logical form — for instance, we will make space for views in which a proposition like *Mary loves John* has at least two logical forms, one involving *loves* applied to these

[2]Russell (1903) appendix B, Myhill (1958). Recent discussions of the paradox and its implications for metaphysics can be found in Dorr (2016), Uzquiano (2015), Goodman (2017), Fritz (2021).

individuals in one order, and another in which *is loved by* is applied in the reverse order. The logical forms give us parameterized classes of propositions with "common structure", but do so in a way that is quite liberal about what one might take "structure" to be. For example, a fairly fine-grained conception of propositions will accept the existence of a propositional function that maps *Aristotle* to the proposition that *Aristotle is wise* and *Socrates* to the proposition that *Socrates is wise*, but they will not accept a function that maps *Aristotle* to the proposition that *Alexander is great* and *Socrates* the the proposition that *there are at least seven planets*. On the other hand, there are more coarse-grained views that accept both functions.[3] Or, could there be a propositional function that maps *Aristotle* to the proposition that *Socrates is wise*, or do all propositional functions preserve the involvement of Aristotle? Even coarse-grained accounts of propositional structure, such as Classicism (see Bacon (2018), Bacon and Dorr (2024)), can admit logical forms with interesting structure, even though one might have initially thought that such views cannot make sense of the notion. This connects with the debate over the existence of vacuous properties, like *being such that Socrates is wise*.[4]

The behaviour of the logical forms will thus tell us a lot about the granularity of propositions. At the coarsest end(s) of the spectrum any function is a logical form and a given proposition has every logical form at once. At the finest, every proposition has a unique logical form. In this way, one can take a class of logical forms satisfying certain closure conditions to be simply a way of *describing* a particular vision about the granularity of reality.

Logical forms are also key to analyzing metaphysical definability: we can say a proposition, like *Aristotle is old and Socrates is wise* is metaphysically analyzed into *Aristotle*, *being old*, *Socrates* and *being wise* if there is a logical form that maps these arguments to this proposition (namely: x is F and y is G).

Before we begin, let me mention a couple of attitudes one might take towards the proposed framework. Russell, Wittgenstein, and others took the question of which propositional functions are *logical forms* to be a substantive question; this position consequently would take questions about what is metaphysically definable from what to be substantive. According to another

[3] An example would be S5 Classicism, described in Bacon and Dorr (2024). For any p and q, the open formula $(x = a \wedge p) \vee (x \neq a \wedge q)$ corresponds to a propositional function that maps a to p and everything else to q.

[4] See Dorr (2016), Goodman (2018) for views that reject such properties.

attitude, however, there is not an important question here. One could take the basic logical operations to be combinators, or to be combinators plus logical words, or to be these plus the primitives of geometry, and each would give us a different notion of metaphysical definability, with none being more theoretically central than the other. While I am inclined towards the former attitude, the framework I am describing here does not presuppose it, and will also be of interest to the pluralist about metaphysical definition.

2 Metaphysical Definability

The central notion of this paper is the notion of 'metaphysical definability'. We will build mathematical models of it, and provide logical systems axiomatizing it. Throughout we use the notation $\overline{t}$ as short for a sequence $t_1, \ldots, t_n$, $\{\overline{t}\}$ for the set $\{t_1, \ldots, t_n\}$, and $\overline{t} : \overline{\rho}$ is short for $t_1 : \rho_1, \ldots, t_n : \rho_n$.

2.1 Languages

To fix ideas, we will start with the different but more familiar notion of linguistic definition. Here we define a very general class of languages which will be both used in explicating linguistic definition, and later as object languages in which to theorize about metaphysical definition. First, some preliminary definitions.

Definition 2.1 (Type assignment)**.** *A type assignment,* Γ*, is a finite sequence* $x_1 : \sigma_1, \ldots, x_n : \sigma_n$ *of variable type pairs, drawn from some infinite stock of variables where no variables are repeated. We abbreviate this sequence as* $\overline{x} : \overline{\sigma}$*.*

We will represent a typed language by a set of *sequents* of the form $\Gamma \vdash M : \sigma$ representing the possible linguistic "propositional functions", when $\sigma = t$, and the analogues of propositional functions for other types σ. The sequence of variables Γ represents the order and types of the arguments the propositional function receives, and M is an open expression of type σ involving variables in Γ, representing the output of the function when the variables are replaced with the argument expressions. In general we write $M : \sigma$ to mean that M is a term of types σ – i.e. there is some sequent of the form $\Gamma \vdash M : \sigma$ in the language. The sequents of the language are specified by a signature, Σ, consisting of a collection Σ^σ of primitive sequents

$\vdash c : \sigma$ for each type σ, and a set of syncategorematic rules, Ξ, generating the remaining terms

$$\frac{\Gamma_1 \vdash A_1 : \sigma_1 \qquad \ldots \qquad \Gamma_n \vdash A_n}{r(\Gamma_1 \ldots \Gamma_n) \vdash r_\Delta(A_1 \ldots A_n) : r(\sigma_1 \ldots \sigma_n)}$$

A rule consists of a mapping from type assignments to type assignments $\Gamma \mapsto r(\Gamma)$ that is insensitive to permutations of variables, and a mapping from finite sequences of types to types $\sigma_1 \ldots \sigma_n \mapsto r(\sigma_1 \ldots \sigma_n)$, as well as an n-ary term forming operation, written $r_\Delta(A_1 \ldots A_n)$ above, where Δ enumerates the *bound variables* the variable type pairs that appear in $\Gamma_1 \ldots \Gamma_n$ but not $r(\Gamma_1 \ldots \Gamma_n)$.

Definition 2.2 (Typed language)**.** *The typed language determined by* Σ *and* Ξ *is the smallest collection of sequents containing* Σ *and closed under the rules in* Ξ*.*

The most familiar kind of typed languages are λ-languages, in which the primary variable binding device is λ-abstraction, and they will be our focus in this paper; although we will at various points indicate how the framework can be generalized to arbitrary languages determined by rules of the above form. A λ-language, $\mathcal{L}(\Lambda, \Xi, \Sigma)$, is given by some set of rules Λ that tell us how to build λ-terms, a set of *simple* syncategorematic rules Ξ than neither bind free variables nor introduce them, and a set of primitive non-logical constants Σ of different types. We will call Λ the λ-*signature*, Ξ the *logical signature* or the *rule signature*, and Σ the *constant signature*. Since the syncategorematic rules $r \in \Xi$ neither bind nor introduce variables (i.e. $r(\Gamma) = \Gamma$ for every type assignment) their syntactic behaviour can be identified with a sequence of input types, $\sigma_1, \ldots, \sigma_n$ and an output type τ: the rule lets us combine expressions of types $\sigma_1, \ldots, \sigma_n$ to form an expression of type τ. We write this $r : \sigma_1, \ldots, \sigma_n \Rightarrow \tau$. We assume that every language contains a syncategorematic rule app : $\sigma \to \tau, \sigma \Rightarrow \tau$, for each pair of types σ and τ, that combines with two expressions of type $(\sigma \to \tau)$ and type σ respectively, F and a, to form an expression of type τ, app(F, a). Crucially, the string 'app' is not itself a meaningful expression of the language; and more generally the notations we choose to represent other syncategorematic operations must not be taken themselves to be meaningful expressions. If we follow the standard convention of writing Fa instead of app(F, a) to represent the application of a predicate to an argument, this confusion is easy to avoid because the

$\frac{\Gamma\vdash M{:}\sigma\to\tau}{\Gamma,x{:}\sigma\vdash Mx{:}\tau}$	**conc**	$\frac{\Gamma\vdash M{:}\sigma\to\tau \quad \Delta\vdash N{:}\sigma}{\Gamma,\Delta\vdash(MN){:}\tau}$	**app**
$\frac{}{\vdash c{:}\sigma}$	**const**	$\frac{r\in\Xi,\Gamma_1\vdash M_1{:}\sigma_1,\ldots,\Gamma_n\vdash M_n{:}\sigma_n}{\Gamma_1\ldots\Gamma_n\vdash r(M_1,\ldots,M_n){:}\sigma}$	**rules**

Table 1: Basic rules for constructing propositional functions

syncategorematic notion of application is not notated at all — although this strategy is not applicable to multiple syncategorematic operations without introducing ambiguity into the notation.

Once we have the basic set of rules Ξ and non-logical constants Σ, the sequents of the language are generated by the rules in table 1 (one must keep in mind that these rules are only applicable if the conclusions are sequents, and have no repeated variables on the left).[5] Here the rule "rules" must be understood to respect the input and output types associated with the rule r, and in const $c \in \Sigma^\sigma$. (In this table and henceforth, we adopt the following conventions. By $\overline{\sigma}$ we mean a sequence of types $\sigma_1, \ldots, \sigma_n$. By $\overline{\sigma} \to \tau$ we mean $(\ldots(\sigma_1 \to (\sigma_2 \to (\ldots \to (\sigma_n \to \tau)\ldots)))$, and instead of writing the latter formula we write $\sigma_1 \to \ldots \to \sigma_n \to \tau$, where the brackets are understood to be implicitly associated to the right. By $\overline{x}$, $\overline{A}$ we denote sequence of variables or terms. We will often abbreviate $\lambda x : \sigma.M$ to $\lambda x.M$ when the type σ is clear from context, or irrelevant.) Languages will also have logical vocabulary for stating things, summarized in table 2. The syncategorematic rules uni, cond, and eq allow us to form truth-functional, quantificational and equality claims: we have treated these syncategorematically to remain neutral about the higher-order existence of these operations.[6]

[5]The observant reader may notice that in the statement of merge, we have merged the variable to the left, as opposed to the right. There is a question here about the appropriateness of such rules given different metaphysical theories of relations, but ultimately boils down to a syntactic convention. See the discussion of left and right contraction in Bacon (2023b) p259-260.

[6]In the presence of cond and abs we can form a conjunction connective $\lambda p\lambda q.(p \wedge q)$. Our treatment is not *entirely* neutral, because we have by fiat ruled out the possibility that abstraction is not available in general, but that there exists a conjunction connective. There is no reason our treatment wouldn't extend to this sort of language, but the generality it would afford us to do so does not justify all the extra cases we would have to consider in all our proofs.

$\dfrac{\Gamma\vdash A{:}t \qquad \Delta\vdash B{:}t}{\Gamma,\Delta\vdash(A\to B){:}t}$	**cond**	$\dfrac{\Gamma\vdash A{:}\sigma \qquad \Delta\vdash B{:}\sigma}{\Gamma,\Delta\vdash(A=_\sigma B){:}t}$	**eq**
$\dfrac{\Gamma\vdash A{:}t}{\Gamma\backslash\overline{x}\vdash\forall y.A[y/\overline{x}]}$	**uni**		

Table 2: Rules for constructing logical propositional functions

$\dfrac{}{x{:}\sigma\vdash x{:}\sigma}$	**id**	$\dfrac{\Gamma,x{:}\sigma,y{:}\tau,\Delta\vdash M{:}\rho}{\Gamma,y{:}\tau,x{:}\sigma,\Delta\vdash M{:}\rho}$	**switch**
$\dfrac{\Gamma,\Delta\vdash M{:}\tau}{\Gamma,x{:}\sigma,\Delta\vdash M{:}\tau}$	**vac**	$\dfrac{\Gamma,x{:}\sigma\vdash M{:}\tau}{\Gamma\vdash\lambda x{:}\sigma.M{:}\sigma\to\tau}$	**abs**
$\dfrac{\Gamma,x{:}\sigma,\Delta,y{:}\sigma,\Lambda\vdash M{:}\tau}{\Gamma,z{:}\sigma,\Delta,\Lambda\vdash M[z/x][z/y]{:}\tau}$	**merge**		

Table 3: Structural rules for a λ signature

Since formulas involving logical notions can all be defined up to extension in terms of $\to, \forall_\sigma$ and $=^\sigma$, we take only these as primitive to save us some cases in inductive proofs, although there is no reason we couldn't have included primitive rules for existentials, conjunction, and so on.(For technical reasons we will occasionally consider languages that do not contain logical propositional functions.)

Finally, the propositional functions of a language may be closed under certain structural operations: **switch**, **vac**, **merge**, **id** and **abs**, summarized in table 3. These are optional, and can be added or removed in order to control which sorts of λ terms are formulable. The λ-signature $\Lambda \subseteq \{\mathbf{switch}, \mathbf{vac}, \mathbf{merge}, \mathbf{id}, \mathbf{abs}\}$ determines which of the five optional rules are included. A 0-ary syncategorematic rule in Ξ has no input expression and outputs a term of type σ, and so functions like a constant — we will call these *logical* constants, in distinction from the non-logical constants that are members of Σ.

Definition 2.3. *Let* $\Lambda \subseteq \{\mathbf{switch}, \mathbf{vac}, \mathbf{merge}, \mathbf{id}, \mathbf{abs}\}$. *By* $\mathcal{L}(\Lambda, \Xi, \Sigma)$ *we mean the language obtained by the rules in table 1, table 2, the* λ*-rules in* Λ *from table 3, syncategorematic rules* Ξ *and non-logical constants* Σ. *By*

$\mathcal{L}^{-}(\Lambda, \Xi, \Sigma)$ *we will mean the same minus the rules for logical operations in table 2.*

All our languages have the syncategorematic operation of application. But why must we have any syncategorematic rules at all? Couldn't we just have, say, a constant app: $(\sigma \to \tau) \to \sigma \to \tau$ to do the job of application? The answer is of course no: how would we combine app with its arguments without a syncategorematic rule? Syncategorematic rules are unavoidable. Moreover, symbols taken to indicate syncategorematic operations should not be mistaken for meaningful expressions in their own right. This is what Russell says about the 'is' of predication. Russell's view was the the word 'is' in 'Socrates is wise' does not itself contribute a constituent to the proposition expressed by this sentence — it is genuinely syncategorematic. (This sentiment can be articulated without committing ourselves to Russell's structured view of propositions by simply saying that the word 'is' is not meaningful expression on its own, but a word that indicates how the predicate and name are combined.) The alternative picture is that 'is' is a further constant, a binary relational expression, and the logical form of *Socrates is wise* is more complicated: *Socrates* bears the instantiation relation to *being wise*. Russell rejects this on the grounds that a regress looms: if this proposition has *Socrates* and *being wise* instantiating the instantiation relation, there would need to be an even higher-order instantiation relation to hold them together, and so on (this is related to Bradley's regress, Bradley (1893)). We need, at some point, a syncategorematic way of gluing predicates to their arguements. Thus it seems that any language that contains simple predications like 'Socrates is wise' should have a syncategorematic rule for application as primitive, or as somehow defined.

The presence of the rule **abs** ensures that it is always possible to introduce definitions by *abstraction*. An open formula in one variable, $x : e \vdash A : t$, can be thought of a set of closed sentences parameterized by the names in the language—a *propositional function*—differing from one another only in which name takes the position of x. For instance: *Socrates is wise and Socrates is old*, *Aristotle is wise and Aristotle is old*, ... If definitions by abstraction are permissible, we can, for every parameterized class of sentences (i.e. propositional function), augment the language with a new interpreted predicate, F (in this case, plausibly *is old and wise*) with the property that when every you apply F to a name a the result Fa is synonymous with the sentence in our class corresponding to that name, $A[a/x]$. Observe that without the

rule abs it is not possible to form *any* λ terms, and further choices about which of the other optional rules, like **switch** and **merge**, to adopt, will not make a difference to which closed expressions exist in the language. However, these choices *will* make a difference if other variable binding devices are added to the language and, crucially, to which "propositional functions" (i.e. sequents) there are. Languages with different open sequents have a different structure, and may differ concerning what is (linguistically) definable from what, even if they agree about which closed terms there are. Since we are identifying a language with its set of propositional functions rather than the set of closed expressions, two distinct languages can have the same closed expressions. There are languages intermediate between those containing the unrestricted **abs** rule, and those completely lacking it: one could introduce a condition on the formation of λ-terms, such as the condition that one cannot bind into the scope of a syncategorematic rule. These intermediate languages can be modeled in our framework, but for reasons of space we will not treat them directly.

To illustrate, let's consider some examples of languages with different syncategorematic rules. According to the logical atomists (Wittgenstein (1921), Russell (1940), Ramsey (1999)), logical words such as 'and' and 'not' are like 'is'. There is no such thing as an operation of conjunction that is a constituent of a proposition $A \wedge B$—conjunction is a syncategorematic operation like application, that directly combines with A and B to form $A \wedge B$, in exactly the same way that predicate and argument are directly combined.[7] The $\wedge$ symbol is just punctuation, like juxtaposition is for application, and doesn't denote a propositional constituent. Thus conjunction, for instance, should be treated as syncategorematic: $\wedge : t, t \Rightarrow t \in \Xi$. But in order to prevent us reintroducing a defined conjunction connective, we must modify the Λ signature. For if **id** and **abs** are both in Λ, we can obtain a conjunction

[7]Why might one be tempted to adopt the logical atomist picture of logical constants? One argument comes from the idea that logic is what is true purely in virtue of logical form: what is true no matter which constituents are replaced. If conjunction, disjunction, and so on, were logical constituents and could be replaced there would be no logical laws in the propositional calculus.

expression as follows:

$$\dfrac{\dfrac{\dfrac{p : t \vdash p : t \qquad q : t \vdash q : t}{p : t, q : t \vdash p \wedge q : t}}{p : t \vdash \lambda q.p \wedge q : t}}{\vdash \lambda pq.p \wedge q : t}$$

It is easy to see that removing **id** blocks the formation of a conjunction connective, but allows things like the property conjunction, $\lambda xy.(Fx \wedge Gy)$, of two predicates $F, G : e \to t$. Removing **abs** blocks the formation of all λ-terms, but then one could mitigate this by introducing a restriction of **abs**, that permits λ abstraction when binding happens outside the scope of a conjunction. At any rate the first strategy, which is treated straightforwardly in our framework, provides us with our first example of a language:

Example 2.1 (Logical atomism)**.** *The logical atomist language is the language* $\mathcal{L}(\Lambda, \Xi, \Sigma)$ *where* $\Lambda = \{\mathbf{switch}, \mathbf{vac}, \mathbf{merge}, \mathbf{abs}\}$ *and* Ξ *contains the syncategorematic rules:*

$$app^{\sigma\tau} : \sigma \to \tau, \sigma \Rightarrow \tau$$

$$\wedge : t, t \Rightarrow t$$

$$\vee : t, t \Rightarrow t$$

$$\neg : t \Rightarrow t$$

In this example we have set aside the logical atomists syncategorematic treatment of the quantifiers, which we will come to later.

Consider next the "positionalist" view about relations (Fine (2000)). Many philosophers have expressed scepticism about the existence of converse relations.[8] According to the positionalist about relations, there aren't two relations, *loves*, and *is loved by*, but a single loving relation with two positions into which two arguments can be plugged in two different ways. The positionalist therefore will want to reject a language that lets one construct two distinct relational expressions of this sort.

[8]Fine (2000), Williamson (1985), Dorr (2004) for relevant discussion.

Given the rule switch it is possible to generate a converse to any relation, L, as follows

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash L : e \to e \to t}{x : e \vdash Lx : e \to t}}{x : e, y : e \vdash Lxy : t}}{y : e, x : e \vdash Lxy : t}}{y : e \vdash \lambda x.Lxy : t}}{\vdash \lambda y \lambda x.Lxy : t}$$

So the positionalist might want to work in a λ-signature without switch. Similar considerations might push one to reject the existence of reflexized unary properties, like *loves oneself*, in favour of a single binary relation in which the same argument can be supplied twice, or to reject vacuous properties like *being such that snow is white*.

Example 2.2 (Positionalism)**.** *The basic positionalist language is the language* $\mathcal{L}(\Lambda, \Xi, \Sigma)$ *where* $\Lambda = \{\mathbf{abs}, \mathbf{id}, \mathbf{merge}, \mathbf{vac}\}$ *and* Σ *contains a binary predicate* L *(*$L :\Rightarrow e \to e \to t$*).*

These two examples illustrate the sort of generality that our treatment of languages has. It should be noted that all the languages we have described not only contain application, but also composition as a defined syncategorematic notion, like $\vdash \lambda p.(\neg(\Box(\neg p)))$, and a more general mode of combination that simultaneously generalizes application and composition called *complication* (Bacon (2023a)). One could relax even this by treating treating variable assignments as built up by a possibly non-associative operation of combining pairs of variable assignments, but we will not explore this level of generality here.[9]

2.2 Linguistic definition

In this section we provide a suggestive model for metaphysical definition, the linguistic notion of definability. Key to this is the notion of a logical form and a propositional function, given a language $\mathcal{L}(\Lambda, \Xi, \Sigma)$. Again, the notion plays a leading role in the logical atomists' philosophy of logic. Take the proposition *Socrates was before Aristotle*. The constituents of the proposition are *Socrates*, *Aristotle* and the relation *before*. The proposition also has

[9]See the discussion in Bacon (2023b) p209 and 221, Restall (2000).

a logical form, which tells us how these constituents are put together. Russell defines it as what is common to all propositions obtained by uniformly substituting the constituents of a proposition for other constituents of the same type. In this case the form may be represented as 'x Rs y' – it is the instantiation of a relation by two arguments.

Let $x_1 : \sigma_1 \ldots x_n : \sigma_n \vdash C : \tau$ be a sequent of a language $\mathcal{L}(\Lambda, \Xi, \Sigma)$. For any sequence of sequents $\Gamma_i \vdash A_i : \sigma_i$ for $i = 1 \ldots n$, we write $\overline{\Gamma} \vdash C[\overline{A}/\overline{x}]$ for the sequent obtained by replacing each occurrence of x_i with A_i in C. The Substitution Lemma, proven in the appendix, ensures that this is also a sequent of $\mathcal{L}(\Lambda, \Xi, \Sigma)$. Following Russell's idea we reach the following definition of a logical form, and a propositional function.

Definition 2.4 (Propositional functions and logical forms)**.** *Suppose that $x_1 : \sigma_1 \ldots x_n : \sigma_n \vdash C : \tau$ is a sequent of the language $\mathcal{L}(\Lambda, \Xi, \Sigma)$. The Russellian function corresponding to this sequent is the function* $\mathbf{r}$ *defined as follows:*

$$\mathbf{r} : \mathcal{L}^{\sigma_1}(\Lambda, \Xi, \Sigma) \times \ldots \times \mathcal{L}^{\sigma_k}(\Lambda, \Xi, \Sigma) \to \mathcal{L}^{\tau}(\Lambda, \Xi, \Sigma)$$

$$\mathbf{r}(A_1, \ldots, A_n) = C[\overline{A}/\overline{x}].$$

We say that $\mathbf{r}$ *is a Russellian function in the constants $\{c_1, \ldots, c_n\}$ iff if $x_1 : \sigma_1 \ldots x_n : \sigma_n \vdash C : \tau$ is a sequent of the language $\mathcal{L}(\Lambda, \Xi, \{c_1, \ldots, c_n\})$ — i.e. the only constants in C are $c_1 \ldots c_n$. We say that* $\mathbf{r}$ *is the* logical form *of C iff C has no constants — i.e.* $\mathbf{r}$ *is a Russellian function in $\mathcal{L}(\Lambda, \Xi, \emptyset)$. If C has type t then we call the corresponding notions a* propositional logical form*, and* propositional function*.*

One can imagine a propositional function as the result of "punching out" some of the occurrences of constants (variables), in a formula A, to yield a formula with "holes" that can be filled in different ways with other expressions of appropriate types to form the result of applying the propositional function to those expressions. A Russellian function is defined in the same way except that A can be an expression of any type. A logical form is a propositional function in which *all* the constants have been punched out leaving only the most general form of the expression. NB: if the language contains **switch**, there actually two distinct logical forms corresponding to a single term like Rxy, namely the the logical forms given by $x : \sigma, y : \tau \vdash Rxy : t$, and $y : \tau, x : \sigma \vdash Rxy : t$. These logical forms are very unfamiliar from a structured point of view, but our goal here is not to capture the structured

view specifically, but to develop a framework that is neutral about the nature of propositional structure and logical form.

Let us now discuss the notion of linguistic definability. There are, of course, purely syntactic notions of definability — relations between uninterpreted strings — that one could study. Our target, however, is a relation between interpreted expressions of an interpreted language. An interpreted language comes equipped with a notion of truth simpliciter that applies to sentences of the language, and a relation of synonymy between (possibly open) expressions of the same type, governed by the following:

- Synonymy is an equivalence relation.
- Synonymous terms may be substituted for one another while preserving synonymy, additionally preserving truth if the target of the substitution is a formula.

Notice, in particular, that I have allowed the relation of synonymy to hold between *open* expressions, and that open expressions may be substituted for each other even in contexts in which free variables in those expressions get bound. Thus, if 'x is a bachelor' is synonymous with 'x is a man and x is not married', then '$\exists x.x$ is a bachelor' is synonymous with '$\exists x.x$ is a man and x is not married'.

According to this operative class of notions, the interpreted predicate 'vixen' can be defined from 'female' and 'fox' on the assumption that 'vixen' is synonymous with 'female fox', even though there is no obvious purely syntactic connection of definability between these strings. In what Russell calls a logically perfect language, one would not have simple expressions, like 'vixen', corresponding to logically complex properties — language would reflect the world in the sense that the simple expressions (constants) correspond one to one with the simple, or fundamental entities. But even in logically perfect languages we might still have non-trivial relations of synonymy between *complex* expressions. We will assume, minimally, that $\beta\eta$-equivalent expressions (open or closed) are synonymous, and leave open what further instances of synonymy between complex expressions obtain (a natural strengthening is that *logically* equivalent terms are synonymous[10]). Thus, for instance,

[10]This in essence delivers an interpreted language in which *Classicism* holds. Assume that substitution of synonymous terms preserves truth, even in contexts where free variables in the terms being substituted can get bound. Whenever $A \leftrightarrow B$ is a logical truth,

$\lambda x.(Fx \wedge Gx)a$ is synonymous with $Fa \wedge Ga$.[11]

With notions of logical form and propositional function at hand we may introduce various different notions of linguistic definability.

Definition 2.5 (Logical definability.)**.** *A sequence of (possibly open) terms $\overline{A} = A_1, \ldots, A_n$, logically defines another, B iff there exists a logical form* $\mathbf{r}$ *such that* $\mathbf{r}(A_1, \ldots, A_n)$ *is synonymous with B.*

(when $\overline{A}$ and B are open expressions, the synonymy of $r(\overline{A})$ with B relies on our notion of synonymy for open terms.)

For logical definability, the order and number of occurrences of an element in $\overline{A}$ may matter: in some (but not all) languages, the sequence 'Mary', 'loves', 'John' defines 'Mary loves John', but not 'John loves Mary'. This kind of definition is possible only if there is a propositional function $X : e \to e \to t, y : e, z : e \vdash A[X, y, z] : t$ in the language such that $A[L/X, m/y, j/x]$ is synonymous with Ljm, using L, m and j for 'loves', 'Mary' and 'John'. In languages that contain id and switch, in particular, we have the sequent $X : e \to e \to t, y : e, z : e \vdash Xzy : t$ that serves this role.

Similarly, in some interpreted languages the sequence 'Mary', 'loves' defines 'Mary loves' but not 'Mary loves Mary'. In languages that contain **merge**, there is a propositional function $X : e \to e \to t, y : e \vdash Xyy : t$ which has the feature that it maps L and m to Lmm. Lastly, in some languages every element in $\overline{A}$ must be used in a definition: thus 'Mary', 'loves', 'John' does not logically define 'Mary loves'. But if the language contains **vac** then we have the propositional function $X : e \to e \to t, y : e, z : e \vdash Xy$ which permits this definition.

These languages differ over which propositional functions are available, and so are giving us different models of linguistic definition and propositional structure. In some languages things like order and constituent count do not matter at all. These will contain the logical form corresponding to '... loves ...' in addition to the logical form '... loves ___' — in the latter there are two gaps, which can be filled with different names, and in the former there is only one and putting a name into that gap produces a sentence with two occurrences of the name.

---

then $C \leftrightarrow C[A/B]$ must be true (it must have the same truth value as $C \leftrightarrow C$. And Classicism can be axiomatized by all truths of this form (see Bacon and Dorr (2024)).

[11]The notion of $\beta\eta$-equivalence is standard – a definition can be found in Bacon (2023b) §3.3, definitions 3.5 and 3.6.

Note, finally, that there is an important difference between a propositional function with gaps $\overline{\rho}$ and a relation of type $\overline{\rho} \to t$. The kind of Tractarian that rejects Abs will accept the propositional function $p : t \vdash \neg p : t$ but not the existence of a unary operator $\vdash \neg : t \to t$. One might have thought that B is logically definable from $A_1, \ldots, A_n$ if there is a closed logical n-ary predicate P such that $PA_1 \ldots A_n$ is synonymous with B, but this is not so in general (cf Bacon (2020) §3.3 remark 3.1). For the Tractarian we just described *Socrates is not wise* is definable from *Socrates is wise*, but there is no logical operation mapping the latter to the former. It is only in the presence of Abs (and the synonymy of $\beta\eta$-equivalents) that propositional functions are equivalent to closed expressions, and that such a reduction would be possible; see the principle Decomposition discussed in section 5.2.

Logical definability corresponds to the idea that we can define one thing from another in strictly logical terms — just using the syncategorematic operations of the language. But we might also wish to formulate, for example, the claim that mental predicates can be defined from some other things in purely physical language: whether there is a *Russellian function* involving only physical constants that lets us define mental properties from, say, names of particular physical systems.

Definition 2.6 (Conditional definability.)**.** *A sequence of terms $\overline{A}$ defines C conditional on the sequence $\overline{B}$ when there is a propositional function* $\mathbf{r}$ *in the elements $\overline{B}$ (i.e. the propositional function involves only elements of $\overline{B}$ in any order any number of times) such that $\mathbf{r}(\overline{A})$ is synonymous with C.*

The rough idea of conditional definability is that C is definable from $\overline{A}$ relative to a language in which the elements of $\overline{B}$ are treated as though they were logical operations in a new language. Observe that logical definability is a special case of conditional definability in which the sequence $\overline{B}$ is the empty. One might have thought that, conversely, conditional definability can be defined in terms of logical definability: $\overline{A}$ conditionally defines C on $\overline{B}$ when $\overline{D}$ logically defines C for some sequence $\overline{D}$ which contains $\overline{A}$ as a subsequence, and whose remaining elements belong to $\overline{B}$. However, in a language without **id**, A does not logically define itself, since the candidate propositional function that consists of just a gap '…', of the same type as A, does not exist; thus in these languages conditional definability must be taken as an additional primitive.

Finally, there is also a notion of priority analogous to that found in Dorr (2016): if it is possible to define one expression from some others using any

expressions whatsoever, we say the latter expressions are prior to the former.

Definition 2.7 (Priority)**.** . *$\overline{A}$ are prior to C when some propositional function maps $\overline{A}$ to C: i.e. C is conditionally definable from $\overline{A}$ conditional on some sequence of terms $\overline{B}$.*

Given these notions, other concepts of interest can be introduced, such as the notion of purity studied in Bacon (2020).

Definition 2.8 (Conditional purity and purity)**.** . *C is pure conditional on the sequence of closed terms $\overline{B}$ iff ϵ (the empty sequence) conditionally defines C on $\overline{B}$. C is pure simpliciter when it is pure conditional on ϵ.*

In languages where $\mathbf{id}, \mathbf{abs} \in \Lambda$, one can construct combinators which are the paradigm examples of pure terms. Examples include the I combinator, $\lambda x.x : \sigma \to \sigma$, C combinator, $\lambda Xyz.Xzy : (\sigma \to \tau \to \rho) \to \tau \to \sigma \to \rho$ and the W combinator, $\lambda Xy.Xyy$.

2.3 Metaphysical definition

Our target is the notion of metaphysical definition, not linguistic definition. While bearing important similarities to the linguistic notion of definition, metaphysical definability is a higher-order relation whose relata are individuals, properties and relations, and so on, as opposed to linguistic items like names and predicates. Having said this, we will often draw on the linguistic notion for intuitions:

Example 2.3. *The property of being a bachelor can be defined from the property of* being a man*, the property of* being married *and negation, $\lambda x.(Mx \wedge \neg Ux)$. The impossibility operation can be defined from the negation operation and the possibility operation ($\lambda p.\neg\Diamond p$). The proposition that* Socrates is wise *can be defined from* Socrates *and* is wise *(Fa).*

While these examples of metaphysical analysis hew closely to the linguistic precedent – application of a predicate to an argument, composition of two operators and so on, some metaphysical definitions are not naturally modeled on any kind of linguistic definition.

Example 2.4. *Perhaps a hydrogen atom may be metaphysically defined, in the operative sense, from the proton and electron that compose it, even if there aren't analogues of linguistic modes of combinations that produce the one from the other.*

If this is so, perhaps the merelogical sum operation, $(\ldots + \_\_\_)$ that has two gaps for individuals, and returns an individual, should count as a metaphysical logical form.

Example 2.5. *A random shape of infinite complexity, such as the coastline of Great Britain, may be metaphysically definable from the Tarski-Hilbert geometric primitives, congruence and betweenness, even if there is no finite linguistic definition of the corresponding shape predicate from the congruence and betweeness predicates. In some sense, once you have* fixed *the facts about the Tarski-Hilbert relations you've fixed the facts about all shapes. See the discussion in Bacon (2023b) [REF].*

In this case, there might be a logical form mapping the two Tarski-Hilbert relations to the property of being Britain-shaped. We will show how one might model these examples in section 4.3.

Another key point is that, unlike the linguistic notions of definition, metaphysical definition has to be taken as primitive. In section 2.2 we defined an expression, C, as logically (linguistically) definable from expressions $B_1, \ldots, B_n$ iff *there exists* a certain logical form — a way of constructing C from $B_1, \ldots, B_n$ using only the syncategorematic operations in the language. The construction of C from $B_1, \ldots, B_n$ can be arbitrarily long, and may use arbitrarily many syncategorematic operations. In higher-order languages it is possible to quantify into different syntactic positions, but it is still not possible to generalize over syncategorematic operations, or constructions. There are three possible ways this may be overcome, but only the last is fully general. The first is available if there is a specific finite number of possible ways to construct something of Cs type from things of the types of $B_1, \ldots, B_n$, for then the existential can be expressed with a finite disjunction. The second is available if all syncategorematic operations "correspond" somehow to something of a given type so that they can, in effect, be quantified over (see the notion of λ-closure in section 3.3, and the corresponding axiom Abs in section 5.2). For instance, with **id** and **abs** there is a term $\lambda X \lambda y.Xy : (\sigma \to \tau) \to \sigma \to \tau$ corresponding to the syncategorematic notion of application. In this case, one instead needs a primitive predicate saying which entities "correspond" to syncategorematic operations. If neither of these cases obtain it is not possible to provide an explicit definition of an analogous notion of metaphysical definability in a higher-order language — we cannot quantify over the "possible ways" of constructing C. The third way, the approach adopted here, is instead axiomatic. We will take the

notion of metaphysical definability as primitive and provide principles governing it that entirely capture the intended notion. Our theorizing about this primitive may be guided by the pretense that it is possible to quantify over syncategorematic operations, but this should be taken with a pinch of salt—the axioms must ultimately be evaluated on their own terms. For each type σ and sequence of types $\overline{\rho} = \rho_1, \ldots, \rho_n$, we need a further pair of variable binding syncategorematic devices $\rhd^{\overline{\rho}\sigma}_{\overline{x}}$, and $\preceq^{\sigma\tau}_{\overline{x}}$

- $B_1 \ldots B_n \rhd^{\overline{\rho}\sigma}_{x_1 \ldots x_k} C$ says that entities $B_1, \ldots, B_n$ (of types $\rho_1, \ldots, \rho_n$) metaphysically define C.
- $B_1 \ldots B_n \rhd^{A_1 \ldots A_m \overline{\rho}\sigma}_{x_1 \ldots x_k} C$ says that entities $B_1, \ldots, B_n$ (of types $\rho_1, \ldots, \rho_n$) conditionally metaphysically define C conditional on $A_1, \ldots, A_m$.
- $B \preceq^{\sigma\tau}_{\overline{x}} C$ says that B is metaphysically prior to C.

When $B_1 \ldots B_n$ and C are open and contain any of the variables $x_1 \ldots x_k$ free, we can think of these terms as Russellian functions in these variables. These variables get bound in the corresponding definability claims, and we should understand them as saying that the potential propositional function $C[\overline{x}]$ can be defined from (is prior to) the propositional functions $B_1[\overline{x}] \ldots B_n[\overline{x}]$. In model theoretic terms this means there is a propositional function such that, maps $B_1 \ldots B_n$ to C under all possible values of $\overline{x}$. (The linguistic analogue of this feature is that our notion of definability employed a notion of synonymy between open term.)

These expressions can be added to the language by the rules in table 4 in the case that $\mathbf{id} \in \Lambda$. In each case, we understand that $\overline{x}$ contains only free variables appearing in $A_1, \ldots, A_n, B$, and in the last case, no free variables appearing in $C_1 \ldots C_m$. In the case that $\mathbf{id} \notin \Lambda$ we need a slightly more complicated statement of the rule, that is equivalent to the above when $\mathbf{id} \in \Lambda$. This is because we would like a free variable to appear on its own as an argument of a definability claim — such as in the claim $x \rhd^{\sigma}_{x} x$ – and this is not obtainable from the above rule without **id** since the variable x is not a term.

$$\frac{\Gamma_1 \vdash A_1 \qquad \ldots \qquad \Gamma_n \vdash A_n \qquad \Sigma \vdash B}{\Delta \vdash t_1 \ldots t_k \rhd_{\overline{x}} B}$$

where Δ is the sequence $s_1 \ldots s_k$ where s_i is either a variable or the sequence of variables appearing in Γ_j, and t_i is s_i if s_i is a variable, and t_i is A_i if s_i is Γ_i. In either case, we call the resulting language $\mathcal{L}^{+}(\Lambda, \Xi, \Sigma)$.

$\frac{\Gamma_1 \vdash A_1{:}\sigma_1 \ldots \Gamma_n \vdash A_n{:}\sigma_n \; \Delta \vdash B{:}\tau}{\Gamma_1 \ldots \Gamma_n \Delta \backslash \overline{x} \vdash A_1 \ldots A_n \rhd_{\overline{x}}^{\overline{\sigma}\tau} B{:}t}$	**Def**	$\frac{\Gamma \vdash A{:}\sigma \; \overline{\Delta} \vdash B{:}\tau}{\Gamma, \Delta \backslash \overline{x} \vdash A \preceq_{\overline{x}}^{\sigma\tau} B{:}T}$	**Prior**
$\frac{\Gamma_1 \vdash A_1{:}\sigma_1 \ldots \Gamma_n \vdash A_n{:}\sigma_n \; \Sigma_1 \vdash C_1{:}\tau_1 \ldots \Sigma_m \vdash C_m{:}\tau_m \; \Delta \vdash B{:}\tau}{\Gamma_1 \ldots \Gamma_n \Delta \Sigma_1 \ldots \Sigma_m \backslash \overline{x} \vdash A_1 \ldots A_n \rhd_{\overline{x}}^{C_1 \ldots C_m \overline{\sigma}\tau} B{:}t}$	**CDef**	$\overline{x}$ not free in $\overline{C}$	

Table 4: Formation rules for definability primitives

A final point on our primitive of conditional definability. One might have thought there is a more general notion of propositional function: a propositional function constructable using the logical forms along with entities *and* a number of propositional functions.

> $B_1 \ldots B_n \rhd_{\overline{x}\overline{y}_1 \ldots \overline{y}_m}^{A_1 \ldots A_m \overline{\rho}\sigma} C$ says that entities $B_1, \ldots, B_n$ (of types $\rho_1, \ldots, \rho_n$) conditionally metaphysically define C conditional on the propositional functions $\overline{y}_1 : \overline{\rho}_1 \vdash A_1, \ldots, \overline{y}_m : \overline{\rho}_m \vdash A_m$.

Here $\overline{y}_i$ are a sequence of free variables appearing in A_i and this expression is understood to bind them. This more complicated primitive can easily modelled in the present framework. However, my view is that this primitive does not add important expressive power to our theory: whatever can be defined from $\overline{B}$ conditional on some further propositional functions, $\mathbf{r}_1 \ldots \mathbf{r}_n$, can be defined from $\overline{B}$ conditional on some entities (i.e. 0-ary propositional functions). This is because all propositional functions, including $\mathbf{r}_1, \ldots, r_n$, can be obtained by filling the gaps in a logical forms with constituents, so these constituents conditionally define anything we can define with those propositional functions.

Could there be propositional functions that are neither logical forms, nor obtainable from them by filling in the gaps of a logical form with constituents? I think not: if we start off with a list of logical forms and encounter a propositional function that cannot be constructed by filling the gaps of that logical form with constituents, then I think we should say that our original list was incomplete and that we have discovered a new logical form.

3 Models of Metaphysical Definability

Despite being unable to explicitly articulate what it takes for one entity to be metaphysically definable from some others, we can still test our axiom system for correctness by supplying it with an intended model theory modeled on the linguistic notion of definability — a model theory that *does* involve quantification over all the possible Russellian functions — and proving soundness and completeness of the axiom system with respect to this intended model theory.

In this section we will make heavy use of the following convention for denoting functions.

Definition 3.1. *Given two functions $f : A \to B$, $g : A' \to B'$, we write $f \times g : A \times A' \to B \times B'$ for the function $(a, a') \mapsto (fa, ga')$.*

More generally we write $f_1 \times ... \times f_n : A_1 \times ... \times A_n \to B_1 \times ... \times B_n$ for the function that maps $a_1...a_n \mapsto (f_1a_1, ..., f_na_n)$.

3.1 Definability structures

Given a sequence $\overline{\rho} = \rho_1, \ldots, \rho_n$, we write $D^{\overline{\rho}}$ for $D^{\rho_1} \times \ldots \times D^{\rho_n}$, and $A \to B$ for the set of functions from A to B. Here is the key definition of the paper. When ϵ is the empty sequence $D^{\epsilon} = \{*\}$ is a singleton whose only element we will label $*$.

Definition 3.2. *A definability structure is a triple $(D^{\cdot}, R^{\cdot}_{\cdot}, \mathbf{app}^{\cdot})$ where D^{σ} is a set for each type σ, $R^{\sigma}_{\overline{\rho}} \subseteq D^{\overline{\rho}} \to D^{\sigma}$ is a collection representing the logical forms — entities of type σ with "gaps" of types $\overline{\rho}$ — and $\mathbf{app}^{\sigma\tau} : D^{\sigma\to\tau} \times D^{\sigma} \to D^{\tau}$ is the function representing application. A definability structure must be subject to the following constraints*

comp *Whenever $\mathbf{r} \in R^{\tau}_{\overline{\theta}_1\sigma\overline{\theta}_n}$, and $\mathbf{p} \in R^{\sigma}_{\overline{\rho}}$, $\mathbf{r} \circ (\overline{\mathbf{id}}_{\overline{\theta}_1} \times \mathbf{p} \times \overline{\mathbf{id}}_{\overline{\theta}_2}) \in R^{\tau}_{\overline{\theta_1\rho\theta_1}}$*

func *For any logical form $\mathbf{r} \in R^{\sigma\to\tau}_{\overline{\rho}}$, $\mathbf{app} \circ (\mathbf{r} \times \mathbf{id}_{\sigma}) \in R^{\tau}_{\overline{\rho}\sigma}$.*

We call $(D, \mathbf{app})$ the applicative structure underlying *a definability structure $(D, R, \mathbf{app})$.*

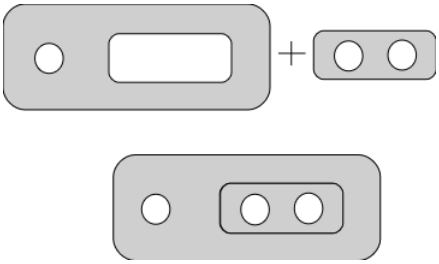
Henceforth we will write D for $\bigcup_{\sigma} D^{\sigma}$ and R for $\bigcup_{\overline{\rho}\sigma} R^{\sigma}_{\overline{\rho}}$ when no confusion arises, and we will sometimes use the following convention:

Convention 3.1. *If* $\Gamma = x_1 : \rho_1, ..., x_n : \rho_n$ *is a context we will write* R^σ_Γ *for* $R^\sigma_{\overline{\rho}}$ *and* D^Γ *for* $D^{\overline{\rho}}$.

Let's begin with some remarks on definition 3.2.

(**comp**) encodes the idea that if you have a Russellian function **r** that has a gap of type σ, and you have another Russellian function **p** of type σ (that is, would yield an entity of type σ when its gaps are filled), then there is another Russellian function obtained by plugging the gappy **p** into the σ gap **r** to make another Russellian function that has the gaps in **r**, but in place of σ gap has **p**, along with its gaps. We can denote this $\mathbf{r} \circ (\overline{\mathbf{id}}_{\bar{\theta}_1} \times \mathbf{p} \times \overline{\mathbf{id}}_{\bar{\theta}_2})$, where $\overline{\mathbf{id}}_{\overline{\pi}} = \mathbf{id}_{\pi_1} \times ... \times \mathbf{id}_{\pi_n}$, $\mathbf{r} \in R^\tau_{\bar{\theta}_1 \sigma \bar{\theta}_n}$, and $\mathbf{p} \in R^\sigma_{\overline{\rho}}$. We will call this operation parallel composition.

As a concrete example, consider the propositional function *x believes that p*, which has two gaps: one for an individual (represented by a circular hole below) one for a proposition (represented by a rectangular hole), and *y loves z* which has two gaps for individual (represented by circular holes). By the operation of parallel composition we get another Russellian function *x believes that y loves z* with three individual holes, represented by the lower diagram below:



For the purposes of illustration we have adopted a diagrammatic representation that suggests that properties and relations are structured entities, however no such assumption is built into the formalism. It is clear that if the Russellian functions are closed under parallel compositions, they are also closed under all precompositions of the form: $\mathbf{r} \circ (\overline{\mathbf{id}} \times \mathbf{r_1} \times \overline{\mathbf{id}} ... \times \overline{\mathbf{id}} \times \mathbf{r_n} \times \overline{\mathbf{id}}) \in R^\sigma_{\overline{\rho}_1 ... \overline{\rho}_n})$.

(**func**) concerns logical forms of simple subject-predicate expressions: if F is a logical form of an entity of type$\sigma \to \tau$, then there is a logical form of entities of type τ obtained by applying F to a gap x of type σ: $Fx : \tau$ is a logical form. The result is a logical form of type τ which has gaps of type $\overline{\rho}$ and now also σ. Mathematically this checks out: $\mathbf{app} \circ (\mathbf{r} \times \mathbf{id})$ is shorthand

for the function $\overline{d}e \mapsto \mathbf{app}(\mathbf{r}(\overline{d}), e)$ — the function that takes a sequence $\overline{d}e$ fills out the logical form of a $\sigma \to \tau$ entity with $\overline{d}$ to obtain an actual entity of type $\sigma \to \tau$, and applies it to e.

It is worth noting that these two conditions, (comp) and (func), entail a stronger condition. Given two logical forms, $\mathbf{r}$ of type $\sigma \to \tau$ and $\mathbf{a}$ of type σ, there's intuitively another logical form obtained by 'applying' $\mathbf{r}$ to $\mathbf{a}$. This is the logical form that has a sequence of gaps matching the gaps in $\mathbf{r}$, another sequence of gaps matching the gaps in $\mathbf{a}$, and when the gaps are filled with entities of the appropriate types, yields the result of filling the gaps in $\mathbf{r}$ and $\mathbf{a}$ with those entities, and applying the result of the former to the latter. However, note that one does *not* need to think of entities as structured with holes in certain orders in order to think that Russellian functions are closed under application in this way.

Definition 3.3 (Complication)**.** *The* complication *of* $\mathbf{r} \in R^{\sigma\to\tau}_{\overline{\rho}_1}$ *and* $\mathbf{a} \in R^{\sigma}_{\overline{\rho}_2}$*, is the function* $\mathbf{app} \circ (\mathbf{r} \times \mathbf{a}) = \overline{d_1 d_2} \mapsto \mathbf{app}(\mathbf{r}(\overline{d}_1), \mathbf{a}(\overline{d}_2))$.

(It is called complication because it is a generalization of composition and application: when $\overline{\rho}_1 = \overline{\rho}_2 = \epsilon$ we have something that amounts to application, and when $\overline{\rho}_1 = \epsilon$ and $\overline{\rho}_2 = \sigma$ we get composition.)

Proposition 3.1. *Definability structures are closed under complication: if* $\mathbf{r} \in R^{\sigma\to\tau}_{\overline{\rho}_1}$ *and* $\mathbf{a} \in R^{\sigma}_{\overline{\rho}_2}$*, then* $\mathbf{app} \circ (\mathbf{r} \times \mathbf{a}) \in R^{\tau}_{\overline{\rho_1\rho_2}}$.

Proof. $\mathbf{r} \in R^{\sigma\to\tau}_{\overline{\rho}_1}$ and $\mathbf{a} \in R^{\sigma}_{\overline{\rho}_2}$. By (func) $\mathbf{app} \circ (\mathbf{r} \times \mathbf{id}) \in R^{\tau}_{\overline{\rho_1\sigma}}$. By (comp), $\mathbf{app} \circ (\mathbf{r} \times \mathbf{id}) \circ (\mathbf{id} \times \mathbf{a}) = \mathbf{app} \circ (\mathbf{r} \times \mathbf{a}) \in R^{\tau}_{\overline{\rho_1\rho_2}}$. □

A final note on some limitations of our framework. We have modeled logical forms (and propositional functions) as functions. This means that if, whenever the gaps in two propositional logical forms are filled with the same constituents the same propositions result, then these are the same logical forms. On a strictly structured view this principle seems obvious: every proposition has a unique logical form, so no two logical forms can yield the same proposition when saturated with *any* constituents, let alone all. However, our framework is neutral on the strictly structured view, and is compatible with views in which a single proposition can have multiple logical forms. For instance, *John loves John* may be decomposed into a logical form involving a binary relation applied to two different arguments or to a single argument twice (compare with the linguistic logical forms $X : e \to$

$e \to t, y : e, z : e \vdash Xyz$ and $X : e \to e \to t, y : e \vdash Xyy$). But what happens if it just so happens that there aren't, say, many individuals around and two distinct logical forms are consequently indistinguistable by any two actually existing individuals. Perhaps we want to say the two logical forms corresponding to a ternary relation, e.g. $x : e, y : e \vdash Rxxy$ and $x : e, y : e \vdash Rxyy$, are different even if there is only one individual, a, and the outputs of these logical forms are always the same, namely $Raaa$. In this case, we want to say that the logical forms are different because it is possible that there be two individuals, and in such a case they would be functionally different. It is thus necessary to revise the framework by weakening the condition of functionality of logical forms. It is actually not too hard to do this by replacing set-theoretic functions with arrows in a "monoidal closed category".[12] However the intended audience for this paper will likely be less familiar with he relevant category theory than the set-theoretic treatment we purue here, so this seems like a reasonable restriction.

Another simplifying assumption I have made is that logical forms are finitary: they are n-ary functions. Someone who thought that propositions could have infinitely many constituents may wish to countenance infinitary logical forms, in which case we would allow $\overline{\rho}$ to range over infinite sequences in our specification, so that $R^{\sigma}_{\overline{\rho}}$ will contain infinite arity functions $\mathbf{r} : D^{\overline{\rho}} \to D^{\sigma}$. Doing so does not change much, although it does slightly effect our discussion of substitution structures in section 4.3.

Lastly, our treatment of logical forms is based on λ-languages, which application as the fundamental syncategorematic operations and optionally λ-abstraction and quantification as the primary variable binders. We might introduce a more general notion of definability structure, for a given signature of arbitrary variable binding operations, that would come, as before, with sets $R^{\sigma}_{\overline{\rho}}$ closed under parallel composition, but also equipped with a collection of manipulations of logical forms, represented by functions s of the form:

$$s : R^{\sigma_1}_{\overline{\rho}_1} \times \ldots \times R^{\sigma_n}_{\overline{\rho}_n} \to R^{\tau}_{\overline{\theta}}$$

where $\overline{\theta}$ is a subsequence of $\overline{\rho}_1 \ldots \overline{\rho}_n$. Note that our notion of a definability structure is given by a single operation on logical forms, $s : R^{\sigma \to \tau}_{\overline{\rho}} \to R^{\tau}_{\overline{\rho}\sigma}$, where $s(\mathbf{r}) = \mathbf{app} \circ (\mathbf{r} \times \mathbf{id})$.

[12] These play an important role in categorical approaches to substructural logics. See Lambek (1968).

3.2 Definability structures for λ-signatures

Let's now turn to definability structures that are appropriate for various λ-signatures $\Lambda \subseteq \{\mathbf{id}, \mathbf{switch}, \mathbf{vac}, \mathbf{merge}\}$ interpreting languages that contain λ-terms formed using the relevant combination of rules from tables 1-3. A structure is appropriate for $\mathbf{id}, \mathbf{switch}, \mathbf{vac}, \mathbf{merge}$ or $\mathbf{abs}$ when it satisfies the condition, to be specified shortly, corresponding to the rule. It is a Λ-definability structure, when it is appropriate for each element of Λ. Here we discuss the conditions for $\mathbf{id}, \mathbf{switch}, \mathbf{vac}$ and $\mathbf{merge}$; we save the condition for $\mathbf{abs}$ until the next section

Definition 3.4. *When* $* \in \{\mathbf{id}, \mathbf{switch}, \mathbf{merge}, \mathbf{vac}\}$*, a* $*$*-definability structure,* $(D, R, \mathbf{app})$ *is defined as follows:*

- *If* $* = \mathbf{id}$ *then* $\mathbf{id}_\sigma \in R^\sigma_\sigma$ *where*

  $$\mathbf{id}_\sigma := a \mapsto a : D^\sigma \to D^\sigma$$

- *If* $* = \mathbf{merge}$ *and* $\mathbf{r} \in R^\tau_{\overline{\rho}_1 \sigma \sigma \overline{\rho}_2}$ *then* $\mathbf{merge}_{\overline{\rho}_1 \sigma \overline{\rho}_2}(\mathbf{r}) \in R^\sigma_{\overline{\rho}_1 \sigma \overline{\rho}_2}$ *where*

  $$\mathbf{merge}_{\overline{\rho}_1 \sigma \overline{\rho}_2}(\mathbf{r}) := \overline{a} b \overline{c} \mapsto \mathbf{r}(\overline{a} b b \overline{c}).$$

- *if* $* = \mathbf{switch}$ *and* $\mathbf{r} \in R^\tau_{\overline{\rho}_1 \sigma \tau \overline{\rho}_2}$ *then* $\mathbf{switch}_{\overline{\rho}_1 \sigma \tau \overline{\rho}_2}(\mathbf{r}) \in R^\tau_{\overline{\rho}_1 \tau \sigma \overline{\rho}_2}$ *where*

  $$\mathbf{switch}_{\overline{\rho}_1 \sigma \tau \overline{\rho}_2}(\mathbf{r}) := \overline{a} b c \overline{d} \mapsto \mathbf{r}(\overline{a} c b \overline{d}).$$

- *If* $* = \mathbf{vac}$ *and* $\mathbf{r} \in R^\tau_{\overline{\rho}_1 \overline{\rho}_2}$, $\mathbf{vac}(\mathbf{r}) \in R^\tau_{\overline{\rho}_1 \sigma \overline{\rho}_2}$ *where*

  $$\mathbf{vac}_{\overline{\rho}_1 \sigma \overline{\rho}_2}(\mathbf{r}) := \overline{a} b \overline{c} \mapsto \mathbf{r}(\overline{a c}).$$

Henceforth we will omit some, or all of the subscripts on $\mathbf{id}, \mathbf{switch}, \mathbf{merge}$ and $\mathbf{vac}$ when no confusion should arise.

These ensure that propositional functions are closed under whatever manipulations on gaps we are allowing. The first condition concerns whether we think that a gap on its own counts as a Russellian function. An affirmative answer is modeled by λ-signatures where $\mathbf{id} \in \Lambda$. (Note that in λ signatures with $\mathbf{id}$ and $\mathbf{abs}$, it is possible to type the identity operation $\vdash \lambda x.x$.) A gap on its own is represented by the identity function $\mathbf{id} : D^\sigma \to D^\sigma$. The remaining conditions ensures the Russellian functions are closed under switching gaps, merging them, and so on, depending on what the λ-signature contains. So, for instance, if there is a propositional function mapping a pair

of elements $a, b \in D^e$ to the element of D^t representing the proposition that *a loves b*, and our theory of propositional function gaps tells us gaps can be switched, then there is another proposition function mapping the pair a, b to the the proposition that *b loves a*.

3.3 λ-closed Definability Structures

In some of the languages we have defined ($\mathcal{L}(\Lambda, \Xi, \Sigma)$ where $\mathbf{abs} \in \Lambda$) there are linguistic logical forms involving λ-abstracts. Here we investigate the metaphysical analogue of these logical forms.

Consider a logical form $\mathbf{p}$ of a proposition that contains a gap for an individual. Does there correspond to this logical form, another logical form, $\lambda\mathbf{p}$, for a property of individuals that has all the same gaps as $\mathbf{p}$ except for the gap for the individual, and which has the feature that if we filled the gaps in $\lambda\mathbf{p}$ and applied the resulting property to an individual we would get the same result as filling the gaps in $\mathbf{p}$ in the same way and filling the final gap with the same individual. More generally, is it true that for every logical form of type τ with gaps of type $\overline{\rho}\sigma$, there is a corresponding logical form of type $\sigma \to \tau$ with gaps $\overline{\rho}$? We can formalize these ideas in the language of definability structures: Suppose $(D, R, \mathbf{app})$ is a definability structure. Given $f : D^{\overline{\rho}\sigma} \to D^\tau$, we define $\lambda f : D^{\overline{\rho}} \to D^{\sigma\to\tau}$ by setting $(\lambda f)\overline{d}$ to be the unique element of $D^{\sigma\to\tau}$ such that $\mathbf{app}((\lambda f)\overline{d}, e) = f(\overline{d}e)$ for every $e \in D^\sigma$ if it exists. λf is undefined otherwise.

Not everyone should answer this question affirmatively. As mentioned previously, the Tractarian account of the truth-functional connectives adopted by many logical atomists might reject this, since the symbol $\neg$, unlike a name or a sentence, doesn't stand for anything in reality — it is more like punctuation.[13] For any proposition there is a further proposition, its negation, but that doesn't mean there's any entity of type $t \to$ out there corresponding to negation. Negation is treated completely analogously to predicate application. Thus there will be a propositional function of negation (linguistically represented by the sequent $p : t \vdash \neg p : t$) obtained by simply punching gaps from a negated proposition, but no corresponding entity of type $t \to t$ (i.e. nothing linguistically represented by $\vdash \lambda p.\neg p : t \to t$).

Definition 3.5 (λ-closure)**.** *Suppose that* $(D, R, \mathbf{app})$ *is a definability struc-*

[13]Classic statements of this view can be found in Wittgenstein (1921), Russell (1940) and Ramsey (1999).

ture, and $R^\sigma_{\overline{\rho}} \subseteq D^{\overline{\rho}} \to D^\sigma$ *for each* $\overline{\rho}$ *and* σ*. Then* $(D, R, \mathbf{app})$ *is* λ*-closed iff:*

> *whenever* $\mathbf{r} \in R^\tau_{\overline{\rho}\sigma}$ *there is a unique function* $\lambda_R\mathbf{r} \in R^{\sigma\to\tau}_{\overline{\rho}}$ *such that* $\mathbf{r}(\bar{d}e) = \mathbf{app}(\lambda_R\mathbf{r}(\bar{d}), e)$ *for every* $\bar{d} \in D^{\overline{\rho}}$*,* $e \in D^\sigma$ *(i.e.* $\mathbf{r} = \mathbf{app} \circ (\lambda_R\mathbf{r} \times \mathbf{id})$*).*

(Note that the partially defined operation λ_R is indexed by a set of functions R and may be well-defined on some sets R and not others. This can happen when R contains too many functions: e.g. if R^σ_σ contain a function $\mathbf{p}$ that does not correspond the applicative behaviour of any element $D^{\sigma\to\sigma}$, $\lambda_R\mathbf{p} \in D^{\sigma\to\sigma}$ will be ill-defined. Or, perhaps $\lambda_R\mathbf{p}$ is well-defined but $\lambda_S\mathbf{p}$ is not well-defined because S contains several functions satisfying the λ-condition when R contains only one. When no ambiguity presents itself, however, we may drop the subscript from λ.)

λ-closure tells us that we can perform the metaphysical analogue of λ-abstraction to logical forms to produce other logical forms. There is, in fact, a stronger condition we might wish to impose: namely that λ-abstraction on arbitrary *propositional functions* is well-defined. A propositional function is a logical form in which entities have been inserted into some of the gaps, and the closure of logical forms under λ-abstraction does not guarantee the closure of propositional functions under the same operation. We will consider this issue in the next section. This further issue will become important when we want to interpret a language $\mathcal{L}(\Lambda, \Xi, \Sigma)$ possibly containing non-logical constants and λ-abstracts in a definability structure.

λ-closure implies the converse of our condition (func)

Proposition 3.2 (Weak λ-closure)**.** *Let* $(D, R, \mathbf{app})$ *be a* λ*-closed definability structure. Then, for any function* $\mathbf{r} : D^{\overline{\rho}} \to D^{\sigma\to\tau}$*, if* $\mathbf{app}\circ(\mathbf{r}\times\mathbf{id}_\sigma) \in R^\tau_{\overline{\rho}\sigma}$ *then* $\mathbf{r} \in R^{\sigma\to\tau}_{\overline{\rho}}$*.*

Proof. uppose $\mathbf{r}$ is an arbitrary function such that $\mathbf{p} := \mathbf{app}\circ(\mathbf{r}\times\mathbf{id}) \in R$ is a logical form. Then $\lambda\mathbf{p} \in R$ is also a logical form, and since $\mathbf{app}\circ(\lambda\mathbf{p}\times\mathbf{id}) = \mathbf{p} = \mathbf{app}\circ(\mathbf{r}\times\mathbf{id}) \in R$ the uniqueness of $\lambda\mathbf{p}$ ensures that $\lambda\mathbf{p} = \mathbf{r}$, so $\mathbf{r}$ is also a logical form. □

A λ-closed definability structure could be defined as one satisfying both (func) and (λ-closure) without including (comp), since the latter can be derived from these two conditions.

Proposition 3.3. *Any structure* $(D, R, \mathbf{app})$ *satisfying (func) and* λ*-closure also satisfies (comp).*

λ-closed definability structures are nice because the logical forms of entities of arbitrary types are determined by the logical forms of propositions and individuals — you only need to specify the base types. Suppose that R and S are two candidate sets of logical forms over the same applicative structure (so we are talking about two definability structures of the form $(D, R, \mathbf{app})$ and $(D, S, \mathbf{app})$). Then we have a conditions for checking when they are the same: they agree on the logical forms of the base types.

Proposition 3.4. *Suppose that* $(D, R, \mathbf{app})$ *and* $(D, S, \mathbf{app})$ *are* λ*-closed structures over the same applicative structure* $(D, \mathbf{app})$*. If for every* $\overline{\rho}$*,* $R^{\kappa}_{\overline{\rho}} = S^{\kappa}_{\overline{\rho}}$ *for base types* $\kappa = e, t$*, then* $R = S$*.*

Thus the logical forms at any type are completely determined by the logical forms of sentences and singular terms. Actually λ-closed structures are also completely specified by what they count as pure; i.e. metaphysically definable from nothing.:

Proposition 3.5. *Suppose that* R *and* S *are both are* λ*-closed, and* $R^{\sigma}_{\epsilon} = S^{\sigma}_{\epsilon}$ *for every type* σ*, then* $R = S$*.*

Proof. In this case the proof is an induction on the length of $\overline{\rho}$. The case where the length of $\overline{\rho}$ is zero is given by assumption. If $\mathbf{r} \in R^{\tau}_{\overline{\rho}\sigma}$ then $\lambda\mathbf{r} \in R^{\sigma\to\tau}_{\overline{\rho}}$ and we can apply the inductive hypothesis, to get that $\lambda\mathbf{r} \in S^{\sigma\to\tau}_{\overline{\rho}}$. (func) then tells us that $\mathbf{app} \circ (\lambda\mathbf{r} \times \mathbf{id}) = \mathbf{r} \in S^{\sigma}_{\overline{\rho}}$. □

Finally, in weakly closed λ-closed structures, if we want to check that a structure satisfies the $\mathbf{merge}, \mathbf{switch}$ or $\mathbf{vac}$ closure conditions we only have to check that the logical forms of propositions and individuals are closed under those conditions.

Proposition 3.6. *If* $R^{t}_{\overline{\rho}}$ *and* $R^{e}_{\overline{\rho}}$ *are closed under* $\mathbf{merge}, \mathbf{switch}$ *or* $\mathbf{vac}$*, then this holds for all types and the the definability structure is closed under* $\mathbf{merge}, \mathbf{switch}$ *or* $\mathbf{vac}$ *respectively.*

Proof. To illustrate the proof strategy we show the case of $\mathbf{merge}_{\rho\theta\rho'}$. It's an induction on the structure of the types, with the base case given by the assumption. Suppose for induction that $R^{\tau}_{\overline{\pi}}$ is closed under merge for any $\overline{\pi}$, and that $\mathbf{r} \in R^{\sigma\to\tau}_{\overline{\rho}\theta\theta\overline{\rho}'}$. By (func) we know that $\mathbf{app} \circ (\mathbf{r} \times \mathbf{id}) \in R^{\tau}_{\overline{\rho}\theta\theta\overline{\rho}'\sigma}$. By

the inductive hypothesis $\mathbf{merge}(\mathbf{app} \circ (\mathbf{r} \times \mathbf{id})) = (a, b) \mapsto \mathbf{app}(\mathbf{r}(a, a), b) = \mathbf{app} \circ (\mathbf{merge}(\mathbf{r}) \times \mathbf{id}) \in R^{\tau}_{\overline{\rho}\theta\overline{\rho}'\sigma}$. Finally by weak λ-closure (proposition 3.2) $\mathbf{merge}(\mathbf{r}) \in R^{\sigma\to\tau}_{\overline{\rho}\theta\overline{\rho}'}$ as required. □

3.4 Conditional Definability Structures

A definability structure tells us which functions are logical forms. A *conditional* definability structure, by contrast, tells us which propositional functions can be constructed from some set of entities. When X is a set of possible constituents, and R the set of logical forms we will write $R[X]$ for the propositional functions with constituents in X.

Given some elements $\overline{d} = d_1...d_n \in D^{\overline{\rho}}$ in some definability structure $(D, R, \mathbf{app})$, there should be another definability structure based on $(D, \mathbf{app})$ of Russellian functions involving $d_1...d_n$. And moreover, there ought to be a "minimal" one, that doesn't involve any more Russellian functions than it needs to. That is, we start of with some functions that represent the *logical forms*: those functions that can be expressed using the syncategorematic operations of some chosen language $\mathcal{L}(\Lambda, \Xi, \Sigma)$ (which correspond to the open expressions of $\mathcal{L}(\Lambda, \Xi, \emptyset)$). Now suppose we want to ask what is metaphysically definable from some geometrical primitives on points of space — say the notions of congruence and betweenness on space-time points? It's natural to consider now an expanded notion of Russellian function: those expressable using the logical syncategorematic operations and the constants of congruence and betweenness (i.e. arbitrary open expressions of $\mathcal{L}(\Lambda, \Xi, \Sigma)$).

There is one more generalization one might be inclined to consider. Might there be new propositional functions that can *only* be obtained by introducing new primitive syncategorematic operations? One picture would be that in addition to the logical syncategorematic forms there are also primitive non-logical ones — perhaps there are geometric forms. But what makes these non-logical forms primitive is that they are not constructable by inserting entities into logical forms. I am not sure that I have a good handle on this picture; the distinction between primitive logical and primitive non-logical forms seems to me to be hard to maintain. Nonetheless, I describe below the machinery needed to make sense.

Given a definability structure $(D, R, \mathbf{app})$ and a collection of functions, $X \subseteq \bigcup_{\overline{\rho}\sigma}(D^{\sigma})^{D^{\overline{\rho}}}$, we will write $R[X]$ for the class of Russellian functions constructed using logical forms and the new rules in X: the "smallest" extension of R in which the elements of X are counted as primitive forms. When we are

taking the position that all propositional functions are obtained by inserting entities into the gaps of logical forms, we restrict X to sets of 0-ary syncategorematic rules: elements of the form $* \mapsto a : D^{\epsilon} \to D^{\sigma}$. Thus $R = R[\emptyset]$ is the class of logical forms. We begin by describing a general class of structures in which the "minimality" of the extension isn't baked in: $R[X]$ is taken to be *any* extension of R containing X that makes $R[X]$ satisfy the coherence conditions for definability structures.

For $a \in D^{\sigma}$ we write $a^* : \{*\} \to D^{\sigma}$ for the function $* \mapsto a$, and for a set of elements from D, X, we write X^* for $\{x^* \mid x \in X\}$.

Definition 3.6 (Conditional Definability Structure)**.** $(D^{\cdot}, R^{\cdot}[\cdot], \mathbf{app})$ *is a conditional* Λ*-definability structure when* $(D^{\cdot}, R^{\cdot}[X], \mathbf{app})$ *is a* Λ*-definability structure for every finite set* $X \subseteq \bigcup_{\sigma}(D^{\sigma})$ *and the following condition holds for any finite sets* $X, Y \subseteq \bigcup_{\sigma}(D^{\sigma})$*:*

$X^* \subseteq R[Y]$ *if and only if* $R[X] \subseteq R[Y]$*.*

The intended model of our notion of conditional definability is thus this. $\Gamma \rhd_{\Sigma} A$ means that the denotation of A (call this a) is obtained by applying a Russellian function constructed from the denotations of the elements of Σ (call these $\overline{c}$) to the sequence elements Γ (call these $\overline{b}$). I.e. for some $\mathbf{r} \in R^{\sigma}_{\overline{\rho}}[\{\overline{c}\}]$, $\mathbf{r}(\overline{b}) = a$.

As we mentioned in section 3.1, our choice to model propositional functions as set-theoretic functions takes away some generality from our framework. This becomes apparent when $\mathbf{abs} \in \Lambda$: λ-closed conditional Λ structures must be functional structures. The reason is simply because *every* functional entity $f \in D^{\sigma \to \tau}$ determines a unary Russellian function $\mathbf{app} \circ (f^* \times \mathbf{id})$ (i.e. $a \mapsto \mathbf{app}(f, a)$). When two such entities, f and g, have the same applicative behaviour these unary Russellian functions are the same and so applying λ to each will result in the same output.[14] Note similarly that while ordinary λ-closed definability structures are not functional in general, they are functional with respect to the logical entities — 0-ary logical forms in $R^{\sigma \to \tau}_{\epsilon}$ for the same reason. I.e. if $f^*, g^* \in R^{\sigma \to \tau}_{\epsilon}$ and f and g have the same applicative behaviour then f and g are identical.

[14] More precisely: suppose $\mathbf{app} \circ (f^* \times \mathbf{id}) = \mathbf{app} \circ (g^* \times \mathbf{id})$. Because $f^*, g^* \in R[f^*, g^*]$, we also have $\mathbf{p} := \mathbf{app} \circ (f^* \times \mathbf{id}) \in R$ id (func). And because $R[f^*, g^*]$ is λ-closed, there is a *unique* element $\lambda\mathbf{p} \in R$ with $\mathbf{app} \circ (\lambda\mathbf{p} \times \mathbf{id}) = \mathbf{p} = \mathbf{app} \circ (f^* \times \mathbf{id})$. Thus $f^* = g^*$ and thus $f = g$.

There are different ways of explicating the notion of a minimal conditional definability structure. We define one of these below: then literal smallest Λ-closed definability structure extending the original containing the new logical entities. But this may not be the only natural way to cash out minimality.

Definition 3.7. *Suppose that* $R^{\sigma}_{\overline{\rho}} \subseteq D^{\overline{\rho}} \to D^{\sigma}$ *is* any *collection of functions. Then we say that* $R[X]$ *is the* Λ*-minimal extension of* R *when it is the intersection of all type indexed collections of functions* $S^{\sigma}_{\overline{\rho}}$ *containing* R *and closed under the following conditions:*

- *If* $\mathbf{r} \in X$ *then* $\mathbf{r} \in S^{\sigma}_{\epsilon}$
- *Whenever* $\mathbf{r} \in S^{\tau}_{\overline{\rho}_1 \sigma \overline{\rho}_3}$ *and* $\mathbf{a} \in S^{\sigma}_{\overline{\rho}_2}$, $\mathbf{r} \circ (\overline{\mathbf{id}} \times \mathbf{a} \times \overline{\mathbf{id}}) \in S^{\tau}_{\overline{\rho_1 \rho_2 \rho_3}}$.
- *Whenever* $\mathbf{r} \in S^{\sigma \to \tau}_{\overline{\rho}}$, $\mathbf{app} \circ (\mathbf{r} \times \mathbf{id}) \in S^{\tau}_{\overline{\rho}\sigma}$.
- *Whenever* $\mathbf{r} \in S^{\sigma}_{\overline{\rho}_1}$, $*\mathbf{r} \in S^{\sigma}_{\rho_2}$ *for each* $* \in \Lambda \subseteq \{\mathbf{vac}, \mathbf{switch}, \mathbf{merge}, \lambda\}$.
- *If* $\mathbf{id} \in \Lambda$, $\mathbf{id} \in S^{\sigma}_{\sigma}$

Given some elements $\overline{d} \in D^{\overline{\rho}}$, *we will write* $R[\overline{d}]$ *for* $R[\{d^*_1, \ldots, d^*_n\}]$.

When $\mathbf{abs} \in \Lambda$, the λ-closure condition may transfer from R to $R[X]$. In this case we call a Λ-definability structure *conditionally* λ-closed with respect to X: the smallest extension $R[X]$ defined above is λ-closed.

3.5 Interpretations

I now specify how to interpret a language $\mathcal{L}(\Lambda, \Xi, \Sigma)$ in a definability structure.

Definition 3.8 (Interpretation)**.** *Let* $(D, R, \mathbf{app})$ *be a definability structure,* Ξ *a signature of syncategorematic operations, and* Σ *a signature of constants. An* interpretation *is a function* $[\![\cdot]\!]$ *with the following*

- *When* $r \in \Xi$ *is a syncategorematic rule, combining with arguments of type* $\overline{\sigma} = \sigma_1, \ldots, \sigma_n$ *to make a term of type* τ, $[\![r]\!] \in R^{\tau}_{\overline{\sigma}}$ *is logical form function.*
- *When* $c \in \Sigma$ *is a non-logical constant,* $[\![c]\!] \in D^{\sigma}$.

When $r \in \Xi$ is 0-ary, i.e. when it is a logical constant, then $[\![r]\!] : D^\epsilon \to D^\sigma$ where $D^\epsilon = \{*\}$ is a singleton set with one element $*$, so that we can think of an interpretation of a constant as simply an element of D^σ for each type.

In some definability structures, the interpretation function can be extended to arbitrary judgments $\Gamma \vdash M : \sigma$ of a given language $\mathcal{L}(\Lambda, \Xi, \Sigma)$, yielding a propositional function mapping the interpretations of the free variables in M to the interpretation of M under that interpretation of the variables: $[\![\Gamma \vdash M : \sigma]\!] : D^\Gamma \to D^\sigma$. In this case we say that the structure satisfies the *enviroment model condition* for that language. Below we repurpose our notation from section 3.3, we write $\lambda[\![\overline{y} : \overline{\rho}, x : \sigma \vdash M : \tau]\!]$ for the unique function $\mathbf{p} : D^{\overline{\rho}\sigma} \to D^\tau$ such that $\mathbf{app} \circ (\mathbf{p} \times \mathbf{id}) = [\![\Gamma, x : \sigma \vdash M : \tau]\!]$ (that is, we are using the partial operation λ_S where $S^\tau_{\overline{\rho}} = (D^\tau)^{D^{\overline{\rho}}}$). This will always exist if the structure is λ-closed conditional on the set of all propositional functions: $X = \{[\![\Gamma \vdash M : \tau]\!] \mid \Gamma \vdash M : \tau \text{ a sequent of } \mathcal{L}(\Lambda, \Xi, \Sigma)\}$.[15]

Definition 3.9 (The environment model condition)**.** $(D, R, \mathbf{app})$ *satisfies the* environment model condition *for the λ-signature Λ if and only if every interpretation $[\![\cdot]\!]$ of signatures Ξ and Σ can be extended to a global interpretation of type sequents of propositional functions satisfying the following constraints:*

Compositionality$_1$ *If* $[\![\Gamma \vdash A]\!] = [\![\Gamma \vdash B]\!]$ *then* $[\![\Delta \vdash C]\!] = [\![\Delta \vdash C[A/B]]\!]$.[16]

Compositionality$_2$ $[\![\Gamma, \Delta, \Gamma' \vdash M[N/x] : \tau]\!] = [\![\Gamma, x : \sigma, \Gamma' \vdash M : \tau]\!] \circ ([\![\Gamma]\!] \times [\![\Delta \vdash N : \sigma]\!] \times [\![\Gamma']\!])$ *where we write* $[\![\Gamma]\!] = [\![x_1 : \sigma_1 \ldots x_n : \sigma_n]\!]$ *to abbreviate* $\mathbf{id}_{\sigma_1} \times \ldots \times \mathbf{id}_{\sigma_n}$.

Application $[\![\Gamma, \Delta \vdash MN : \tau]\!] = \mathbf{app} \circ ([\![\Gamma \vdash M : \sigma \to \tau]\!] \times [\![\Delta \vdash N : \tau]\!])$
(i.e. $\overline{d}_1\overline{d}_2 \mapsto \mathbf{app}([\![\Gamma \vdash M : \sigma \to \tau]\!](\overline{d}_1), [\![\Delta \vdash N : \tau]\!](\overline{d}_2))$*)*

[15] Caution: the λ notation is sensitive to the background collection of functions and may differ between the arbitrary propositional functions, X as defined, and the logical forms R. Observe that it may exist with respect to X but not with respect to R. This can happen when the structure is merely λ-closed and $[\![\Gamma, x : \sigma \vdash M : \tau]\!]$ does not lie in R: unconditional λ-closure only requires logical forms to be abstractable, not arbitrary propositional functions. And even when this function is a logical form in R, a unique $\mathbf{p}$ might exist with respect to X and yet not be in R if R is not λ-closed. Or, if R is λ-closed, it could uniquely exist with respect to R but not uniquely in the whole structure.

[16] This applies even in variable binding contexts; i.e. when C contain variable binders that bind variables in Γ.

Concretion $[\![\Gamma, x : \sigma \vdash Mx : \tau]\!] = \mathbf{app} \circ ([\![\Gamma \vdash M : \sigma \to \tau]\!] \times \mathbf{id}$ *(i.e.* $\overline{d}e \mapsto \mathbf{app}([\![\Gamma \vdash M : \sigma \to \tau]\!], e)$

Rules $[\![\Gamma_1, \ldots \Gamma_n \vdash r(M_1 \ldots M_n) : \tau]\!] = [\![r]\!] \circ ([\![\Gamma_1 \vdash M_1 : \sigma_1]\!] \times \ldots \times [\![\Gamma_n \vdash M_n : \sigma_n]\!])$

Depending on what gap operations belong to Λ*, we require the following*

Abstraction $[\![\Gamma \vdash \lambda x.M : \sigma \to \tau]\!] = \lambda[\![\Gamma, x : \sigma \vdash M : \tau]\!]$ *and the right-hand-side exists.*[17]

Identity $[\![x : \sigma \vdash x : \sigma]\!] = \mathbf{id}_\sigma$

Switch $[\![\Gamma, y : \sigma, x : \tau, \Delta \vdash M : \theta]\!] = \mathbf{switch}[\![\Gamma, x : \sigma, y : \tau, \Delta \vdash M : \theta]\!]$

Merge $[\![\Gamma, z : \sigma, \Delta, \Lambda \vdash M[z/x, z/y] : \tau]\!] = \mathbf{merge}[\![\Gamma, x : \sigma, \Delta, y : \sigma, \Lambda \vdash M : \tau]\!]$

Vac $[\![\Gamma, x : \sigma, \Delta \vdash M : \tau]\!] = \mathbf{vac}[\![\Gamma, \Delta \vdash M : \tau]\!]$

The environment model condition does not give us a recursive definition of the semantic value function $[\![\cdot]\!]$; the existence of a function satisfying these constraints is a condition that a definability structure may or may not satisfy. So how can we guarantee that there are any definability structures satisfying these conditions? This is an important question for practical applications, where one needs to construct concrete models. However in the context of the present paper the answer to this problem is not urgent, so my treatment will be brief.

It is, in fact, possible to turn the above constraints into a recursive definition with the aid of several lemmas. Indeed, in standard treatments of the λ-calculus one starts with the recursive definition and establishes the lemmas (see, for instance, Mitchell (1996) §4.5.3). Observe that the left-hand-side of each of the constraints, apart from Compositionality, is the conclusion of one of the sequent rules, and these are defined in terms of the interpretations of the premises of that same rule on the right-hand-sides of the constraint. One strategy, therefore, is to build the semantic value function up recursively on the structure of the proof of a sequent. (NB: the present approach is thus importantly different than the more standard approach in philosophy and linguistics of building semantic values recursively on the structure of terms.)

[17]

That is, we define a temporary function $[\![\cdot]\!]_D$ that assigns an interpretation to each sequent *relative* to a derivation D, and then we prove that two derivations of the same sequent get assigned the same interpretation:[18]

Proposition 3.7 (Derivation Independence)**.** *If D_1 and D_2 are two derivations of $\Sigma \vdash P : \rho$ then $[\![\Sigma \vdash P : \rho]\!]_{D_1} = [\![\Sigma \vdash P : \rho]\!]_{D_2}$*

This allows us to dispense with the subscripts on $[\![\cdot]\!]$. Then we must prove, again by induction, that the two Compositionality constraints are satisfied. For instance, the second amounts to:

Proposition 3.8 (The Semantic Substitution Lemma)**.** $[\![\Gamma, \Delta, \Gamma' \vdash M[N/x] : \tau]\!] = [\![\Gamma, x : \sigma, \Gamma' \vdash M : \tau]\!] \circ ([\![\Gamma]\!] \times [\![\Delta \vdash N : \sigma]\!] \times [\![\Gamma']\!])$

To give the reader the sense of how this might go, the derivation relative form of the Concretion constraint would be become the following clause of an inductive definition of $[\![\cdot]\!]$ on proof length:

$$[\![\Gamma, x : \sigma \vdash Mx : \tau]\!]_{D;\mathrm{conc}} := \mathbf{app}[\![\Gamma \vdash M : \sigma \to \tau]\!]_D$$

writing $D; C$ for a derivation D followed by a rule C. In general we take each of the identities in definition 3.9 apart from Compositionality and add appropriate subscripts to turn it into a recursive definition on terms not involving $\forall, \to, =, \rhd$ and $\preceq$. In order to state the inductive conditions for universals, conditionals, equality and definability statements we would need operations satisfying certain commutativity conditions. To take universals as our example, we would need a class of operations:

$$\mathbf{uni}_{\overline{\rho}\overline{\rho}'} : (D^{\overline{\rho}} \to D^t) \to D^{\overline{\rho}'} \to D^t \text{ where } \overline{\rho}' \text{ is a subsequence of } \rho.$$

In order to prove the derivation independence of interpretations, we must require that these operations commute with $\mathbf{vac}, \mathbf{switch}$ and $\mathbf{merge}$, in the sense that, for instance, $\mathbf{vac}_\sigma \circ \mathbf{uni}_{\overline{\rho}\overline{\rho}'} = \mathbf{uni}_{\overline{\rho}\overline{\rho}'} \circ \mathbf{vac}_i$, provided σ doesn't appear in $\overline{\rho} \setminus \overline{\rho}'$. And in order to prove the semantic substitution lemma, we must also require that they commute with composition, for instance

[18] This is proved by induction on proof length. Every proof can be factored into the form $D; r, ; C$ where the only rules appearing in C are $\mathbf{vac}, \mathbf{switch}$ or $\mathbf{merge}$, r is some rule other than these rules. One can then shuffle the r forwards over C without disturbing the interpretation, using the fact that each of the operations $\mathbf{vac}, \mathbf{switch}$ or $\mathbf{merge}$ commute with all the other operations used in the semantic clauses in the other rules.

$\mathbf{uni}_{\overline{\rho}\rho'}(\mathbf{r}) \circ (\overline{\mathbf{id}} \times \mathbf{p} \times \overline{\mathbf{id}}) = \mathbf{uni}_{\overline{\rho}''\overline{\rho}'''}(r \circ (\overline{\mathbf{id}} \times p \times \overline{\mathbf{id}})$. Then clause for, say, the Universal sequent rule would be:

$$[\![\Gamma \setminus \overline{x} \vdash \forall y.A[y/\overline{x}] : t]\!]_{D;\mathbf{uni}} = \mathbf{uni}[\![\Gamma \vdash A : t]\!]_D$$

and similar operations must be posited for $\rightarrow$, $\rhd$, $\preceq$ and so on, which we will call **cond**, **def**, **cdef** and **prior**. It is possible then to show by induction the theorem established below, that certain sequents always express logical forms. The details of all of this are unnecessary for present purposes and are omitted for brevity.

Our framework takes the subjects of interpretations to be sequents, $\Gamma \vdash A : \sigma$, and takes these interpretations to be functions whose arguments are intuitively to be thought of as the value of the variables in Γ and whose output is the interpretation of A under those values. A more familiar framework, at least in philosophy and linguistics, takes the subjects of interpretation instead to be terms $A : \sigma$ relative to a variable assignment g: a function mapping each variable type pair, $x : \sigma$, into an element of D^σ. However, these two frameworks are equivalent, and sometimes it is more convenient to switch between them. Given a type assignment $\Gamma = x_1 : \sigma_1 \ldots x_n : \sigma_n$, and g a variable assignment, let us define $g(\Gamma)$ to be the sequence of elements $(g(x_1), \ldots, g(x_n)) \in D^{\sigma_1} \times \ldots \times D^{\sigma_n}$.

Definition 3.10 (Interpretation of a term)**.** *Given a variable assigmnent g, and a term $A : \sigma$ we write $[\![A]\!]^g := [\![\Gamma \vdash A : \sigma]\!]g(\Gamma)$ where $\Gamma \vdash A : \sigma$ is any sequent with A on the right.*

In order for this definition to be well-defined we need to show that this definition doesn't depend on Γ.

Proposition 3.9. *If $\Gamma \vdash A : \sigma$ and $\Delta \vdash A : \sigma$ are two derivable sequents, then $[\![\Gamma \vdash A : \sigma]\!]g(\Gamma) = [\![\Delta \vdash A : \sigma]\!]g(\Delta)$.*

It's easy to see that the structural rules vac, merge and switch leave this value alone: for instance $[\![\Gamma_1, x : \sigma, y : \tau, \Gamma_2 \vdash A : \rho]\!]g\Gamma_1, x : \sigma, y : \tau, \Gamma_2 = [\![\Gamma_1, y : \tau, x : \sigma, \Gamma_2 \vdash A : \rho]\!]g(\Gamma_1, y : \tau, x : \sigma, \Gamma_2 \vdash A : \rho)$. With this fact in hand, the result is easily established by an induction using proposition 3.7. The following lemma, a generalization of Compositionality$_1$, will be useful

later.[19] Write $g[\overline{x}]g'$ to mean that g and g' disagree at most on the variables in $\overline{x}$.

Lemma 3.1. *Let $A, B : \sigma$ have the same variables, A a subexpression of C, and let $\overline{x}$ enumerate the free variables in A that are bound in C. If $[\![A]\!]^{g'} = [\![B]\!]^{g'}$ for all $g'[\overline{x}]g$, then $[\![C]\!]^g = [\![C[A/B]]\!]^g$*

A couple more observations about our definitions should be made at this point. Our definition of the environment condition can easily be extended to languages that have a restricted rule of abstraction — e.g. a language that **abs** to sequents where no free variables appear in the scope of a syncategorematic operation. One simply requires that $\lambda[\![\Gamma, x : \sigma \vdash M : \tau]\!]$ must exist for every λ-abstract, $\Gamma \vdash \lambda x.M : \sigma \to \tau$, *in the language*, whatever they might be. Relatedly, the environment model condition can be satisfied with respect to one type system but not another. When $\Lambda \subseteq \Lambda'$ then every derivable sequent in $\mathcal{L}(\Lambda, \Xi, \Sigma)$ is derivable in $\mathcal{L}(\Lambda', \Xi, \Sigma)$, so that an environment model for Λ' is an environment model for Λ, but not conversely.

Secondly, notice that we have stipulated that an applicative structure satisfies the environment model condition when *every* interpretation of a signature can be extended to all sequents. But for some type systems, if the environment model condition holds for any signature, it holds for every signature. In particular, in type systems containing **id**, there's no difference between where variables and constants can occur: there is a derivation of a sequent $\Gamma \vdash M$, iff there is a way of splitting Γ into $\Gamma_1\Gamma_2$, and there is a derivation of $\Gamma_1, x : \sigma, \Gamma_2 \vdash M' : \tau$ where M' is the result of replacing one occurrence of c with x.[20] So by replacing all the new constants with variables in this manner, we can interpret a sequent not involving the new constants $[\![\Gamma' \vdash M' : \tau]\!]$ appealing to the environment model condition in any signature, and then by filling the appropriate argument places of this function with the interpretations of the relevant constants we obtain an interpretation in the new signature.

[19]The lemma is established by considering a signature containing constants $\overline{c}$ matching the length and types of the sequence of variables $\overline{y}$ obtained by removing the variable $\overline{x}$ from Γ, and an interpretation such that $[\![c_i]\!] = g(y_i)$. One can then apply Compositionality$_2$ to get $[\![\overline{x} \vdash A[\overline{c}/\overline{y}]]\!] = [\![\overline{x} \vdash B[\overline{c}/\overline{y}]]\!]$. Compositionality$_1$ delivers $[\![\Gamma \vdash C]\!] = [\![\Gamma \vdash C[A[\overline{c}/\overline{y}]/B[\overline{cy}]]]\!]$ and using Compositionality$_2$ again, we get the desired result.

[20]Proof sketch of left-to-right: replace the use of Constants at which the relevant occurrence of c is introduced with an application of identity. The right-to-left direction follows from the substitution lemma.

Finally, our framework is general in the following way. While we assume that syncategorematic application is a basic way of constructing new logical forms, we do not build it into our framework that the syncategorematic notions, $\rightarrow$, $\forall_\sigma$, $=_\sigma$, or the definability notions $\rhd$, themselves are logical in the sense that they express logical forms and can be used in the construction of new logical forms. $\rightarrow$, for instance, can be represented by a binary function cond that takes two propositional functions $\mathbf{p} : D^{\overline{\rho}_1} \rightarrow D^t, \mathbf{q} : D^{\overline{\rho}_2} \rightarrow D^t$ and outputs a proposition function $\mathrm{cond}(\mathbf{p}, \mathbf{q}) : D^{\overline{\rho}_1 \overline{\rho}_2} \rightarrow D^t$, but we have not stipulated that cond takes logical forms to logical forms; i.e. that if $\mathbf{p}, \mathbf{q} \in R$ then $\mathrm{cond}(\mathbf{p}, \mathbf{q}) \in R$. It is thus useful to single out a special sublanguage of $\mathcal{L}(\Lambda, \Xi, \Sigma)$, $\mathcal{L}^-(\Lambda, \Xi, \emptyset)$ whose sequents are guaranteed to always express logical forms, namely the language obtained by removing cond, uni, and eq from the rules for $\mathcal{L}(\Lambda, \Xi, \Sigma)$. (Consequently, the sequents of $\mathcal{L}^-(\Lambda, \Xi, \Sigma)$ express propositional functions.)

Proposition 3.10. *For any derivable sequent $\overline{x} : \overline{\rho} \vdash M : \sigma$, of $\mathcal{L}^-(\Lambda, \Xi, \Sigma)$, $[\![\overline{x} : \overline{\rho} \vdash M : \sigma]\!] \in R^\sigma_{\overline{\rho}}[[\![\Sigma]\!]]$ where $[\![\Sigma]\!] = \{[\![c]\!] \mid c \in \Sigma\}$.*

If **uni**, **cond** *and* **eq** *map sequences of elements of R to R, this result extends to $\mathcal{L}(\Lambda, \Xi, \Sigma)$. And if* **def**, **cdef** *and* **prior** *map sequences of elements of R to elements of R then this result additionally extends to $\mathcal{L}^+(\Lambda, \Xi, \Sigma)$.*

3.6 Models

Below we write $g[\overline{x}]g'$ when g and g' agree on all variables except, possibly, on the variables in $\overline{x}$.

Definition 3.11 (Model)**.** *$(D, R, \mathbf{app}, [\![\cdot]\!], v)$ is a model iff $(D, R, \mathbf{app}, [\![\cdot]\!])$ is an interpretation for $\mathcal{L}(\Lambda, \Xi, \Sigma)$ and $v : D^t \rightarrow \{0, 1\}$ is a function satisfying the following constraints, for any variable assignment g:*

- $v([\![A \rightarrow B]\!]^g) = \max(1 - v([\![A]\!]^g), v([\![B]\!]^g))$.
- $v([\![\forall y.A : t]\!]^g) = 1$ *iff* $v([\![A : t]\!]^{g'}) = 1$ *for every variable assignment* $g'[x]g$.
- $v([\![\overline{A} \rhd^{\overline{\rho\sigma}\tau}_{\overline{x}} B]\!]^g) = 1$ *iff there exists some* $\mathbf{r} \in (R^\tau_{\overline{\rho}})$ *such that for all variable assignments* $g'[\overline{x}]g$, $\mathbf{r}([\![A_1]\!]^{g'}, \ldots, [\![A_n]\!]^{g'}) = [\![B]\!]^{g'}]$.
- $v([\![\overline{A} \preceq^{\overline{\rho\sigma}\tau}_{\overline{x}} B]\!]^g) = 1$ *iff there exists some* $X \subseteq D$ *and* $\mathbf{r} \in (R^\tau_{\overline{\rho}}[X])$ *such that for every variable assignment* $g'[\overline{x}]g$, $\mathbf{r}([\![A_1]\!]^{g'}, \ldots, [\![A_n]\!]^{g'}) = [\![B]\!]^{g'}$.

- $v([\![A = B]\!]^g) = 1$ *if and only if* $[\![A]\!]^g = [\![B]\!]^g$.

If $(D, R[\cdot], \mathbf{app})$ *is a conditional definability structure we may also state clause for conditional definability in* $(D, R[\cdot], \mathbf{app}, [\![\cdot]\!], v)$*:*

- $v([\![\overline{A} \rhd_{\overline{x}}^{\overline{C}\overline{\rho}\overline{\sigma}\tau} B]\!]^g) = 1$ *iff there exists some* $\mathbf{r} \in (R_{\overline{\rho}}^{\tau})[[\![C_1]\!]^g, \ldots, [\![C_n]\!]^g]$ *such that for all variable assignments* $g'[\overline{x}]g$, $\mathbf{r}([\![A_1]\!]^{g'}, \ldots, [\![A_n]\!]^{g'}) = [\![B]\!]^{g'}]$.

4 Examples

4.1 The maximal definability structure

The simplest example of a definability structure is one where *every* function counts as a logical form.

Example 4.1 (All functions are logical forms)**.** *Let* $(D, \mathbf{app})$ *be any applicative structure, and suppose that* $R_{\overline{\rho}}^{\sigma}$ *contains all functions* $f : D^{\overline{\rho}} \to D^{\sigma}$*. Then* $(D, R, \mathbf{app})$ *is a* $\{\mathbf{id}, \mathbf{switch}, \mathbf{merge}, \mathbf{vac}\}$*-definability structure.*

It is immediate that R satisfies satisfies the closure conditions on logical forms, (comp) and (func), simply because all the relevant functions are logical forms. For similar reasons it satisfies the the closure conditions for $\{\mathbf{id}, \mathbf{switch}, \mathbf{merge}, \mathbf{vac}\}$.

This structure is not an **abs**-definability structure — i.e. it is not λ-closed — unless for every unary logical form $\mathbf{r} \in R_{\sigma}^{\tau}$, there is a unique function $\lambda\mathbf{r} \in R_{\epsilon}^{\sigma \to \tau}$ such that $\mathbf{app} \circ (\lambda\mathbf{r} \times \mathbf{id}) = \mathbf{r}$ (indeed, this condition must hold for logical forms of arbitrary arity). Now $\lambda\mathbf{r} = * \mapsto d$ for some $d \in D^{\sigma \to \tau}$. So in the present structure, this condition essentially requires that for every function $\mathbf{r} : D^{\sigma} \to D^{\tau}$ there is a unique corresponding element $d \in D^{\sigma \to \tau}$ with the applicative behaviour of that function: $\mathbf{app}(d, a) = \mathbf{r}(a)$ for all $a \in D^{\sigma}$. This means that the applicative structure must be "full" in the sense that every applicative behaviour (i.e. function from D^{σ} to D^{τ}) is realized by some element $d \in D^{\sigma \to \tau}$, and "functional" in the sense that each applicative behaviour is realized by exactly one element of $D^{\sigma \to \tau}$. Every full and functional applicative structure is isomorphic to a "standard" structure in which $D^{\sigma \to \tau}$ consists of all functions from D^{σ} to D^{τ}, and in which **app** denotes function application. Indeed, the converse also holds:in every standard applicative structure, the definability structure consisting of all functions is λ-closed.

Proposition 4.1. *Let* $(D, R, \mathbf{app})$ *be a definability structure where all functions are logical forms (i.e.* $R^{\sigma}_{\overline{\rho}} = (D^{\sigma})^{D^{\overline{\rho}}}$*). Then* $(D, R, \mathbf{app})$ *is* λ*-closed if and only if* $(D, \mathbf{app})$ *is standard (i.e.* $D^{\sigma\to\tau} = (D^{\tau})^{D^{\sigma}}$*.*

The notion of metaphysical definability in the maximal structure is clearly not very interesting. Any model $(D, R, \mathbf{app}, [\![\cdot]\!], v)$ over a maximal definability structure will make all statements of the form $A_1, \ldots, A_n \rhd_{\emptyset} B$ true, since there is always some function or other that maps $[\![A_1]\!] \ldots [\![A_n]\!]$ to $[\![B]\!]$. Note, however, that this does not mean *all* definability claims are true. The previous observation does not extend to claims of the form $A_1, \ldots, A_n \rhd_{\overline{x}} B$ where $\overline{x}$ is a non-empty collection of variables. For instance $\rhd_x Fx$, where $F : \sigma \to \tau$, won't be true unless there is some function $\mathbf{r} : \{*\} \to D^{\sigma\to\tau}$ such that $\mathbf{r}(*) = [\![Fx]\!]^g = [\![Fx]\!]^{g'}$ for every $g[x]g'$. This can only be satisfied if F expresses a vacuous property whose output is the same no matter what its input is; e.g. $\lambda x.A$ where x is not free in A. Indeed, some definability statements, like $\rhd_p(p =_t \top)$, are logically false when $\top$ is treated as a logical constant (i.e. a 0-ary element of Ξ): they are false in all definability models, including the maximal ones. (This logical falsehood should not be confused with the logical truth $p \rhd_p (p =_t \top)$).

4.2 The structure of definable functions

Suppose that we have given an applicative structure $(D, \mathbf{app})$, and we have at interpretation $[\![\cdot]\!]$ of a signature of rules Ξ and constants Σ mapping each rule $r : \overline{\rho} \Rightarrow \tau \in \Xi$ to a function $[\![r]\!] : D^{\overline{\rho}} \to D^{\tau}$, and each constant $c : \sigma$ to an element $[\![c]\!] \in D^{\sigma}$. It's natural to ask whether we can, using $[\![\cdot]\!]$, turn $(D, \mathbf{app})$ into a Λ-definability structure by identifying the logical forms $R^{\tau}_{\overline{\rho}}$ to be just those functions definable from linguistic propositional functions, i.e. sequents $\overline{x} : \overline{\rho} \vdash M : \tau$, of the language $\mathcal{L}(\Lambda, \Xi, \Sigma)$ relative to the interpretation $[\![\cdot]\!]$.

In following Russell, we took our motivating examples of propositional functions and logical forms by analogy with open terms of a language. Thus it is of particular interest to check that the class of functions definable from open terms of a language, suitably interpreted, gives rise to a definability structure. In contrast to the maximal definability structure discussed in the previous section, this is in a natural sense the "smallest" definability structure on an applicative structure that contains the initial interpretations of the primitive rules Ξ and primitive constants Σ.

We say that an *applicative structure* $(D, \mathbf{app})$ satisfies the environment model condition for a λ-signature Λ with respect to an interpretation $[\![\cdot]\!]$ of Ξ and Σ when there is a global interpretation function $[\![\Gamma \vdash A : \sigma]\!]$ on sequents $\Gamma \vdash A : \sigma$ of $\mathcal{L}(\Lambda, \Xi, \Sigma)$ satisfying the conditions from definition 3.9. Observe that while definition 3.9 was given for definability structures, the clauses make sense in an arbitrary applicative structure given an initial interpretation $[\![\cdot]\!]$ of the primitive rules and constants. We will say that the interpretation $(D, \mathbf{app}, [\![\cdot]\!])$ satisfies the environment model condition when $(D, \mathbf{app})$ satisfies it with respect to $[\![\cdot]\!]$.

Example 4.2 (The structure of Λ-definable logical forms)**.** *Suppose that $(D, \mathbf{app}, [\![\cdot]\!])$ satisfies the environment model condition for $\mathcal{L}(\Lambda, \Xi, \Sigma)$. The structure of Λ-definable logical forms over $(D, \mathbf{app}, [\![\cdot]\!])$ is then given by:*

$$R^{\sigma}_{\overline{\rho}} = \{[\![\overline{x} : \overline{\rho} \vdash A : \sigma]\!] \mid \overline{x} : \overline{\rho} \vdash A : \sigma \text{ a sequent of } \mathcal{L}(\Lambda, \Xi, \emptyset)\}.$$

Example 4.3 (The structure of Λ-definable propositional functions)**.** *Suppose that $(D, \mathbf{app}, [\![\cdot]\!])$ satisfies the environment model condition for $\mathcal{L}(\Lambda, \Xi, \Sigma)$ and $[\![\cdot]\!]$ is surjective from Σ^{σ} to D^{σ} at each type σ. The conditional definability structure of Λ-definable propositional functions over $(D, \mathbf{app}, [\![\cdot]\!])$ is then given by:*

$$R^{\sigma}_{\overline{\rho}}[[\![\Sigma_0]\!]] = \{[\![\overline{x} : \overline{\rho} \vdash A : \sigma]\!] \mid \overline{x} : \overline{\rho} \vdash A : \sigma \text{ a sequent of } \mathcal{L}(\Lambda, \Xi, \Sigma_0)\}.$$

where Σ_0 ranges over a finite subsets of Σ

observe that because $[\![\cdot]\!]$ is surjective, every finite subset X of D is of the form $[\![\Sigma_0]\!]$ (i.e. $\{[\![c]\!] \mid c \in \Sigma_0\}$) for some finite set Σ_0.

Proposition 4.2.

1. *The structure of Λ-definable logical forms in $(D, \mathbf{app}, [\![\cdot]\!])$ is a Λ-definability structure.*

2. *The conditional structure of definable propositional functions in $(D, \mathbf{app}, [\![\cdot]\!])$ is a conditional Λ-definability structure; indeed it is the minimal conditional definability structure.*

Proof. Begin with 1. To establish (comp) suppose that $[\![\Gamma, x : \sigma, \Gamma' \vdash A : \tau]\!]$ and $[\![\Delta \vdash B : \sigma]\!]$ are in R. By the substitution lemma $\Gamma, \Delta, \Gamma' \vdash A[B/x] : \tau$ is a sequent of $\mathcal{L}(\Lambda, \Xi[S], \emptyset)$, and so $[\![\Gamma, \Delta, \Gamma' \vdash A[B/x] : \tau]\!]$ belongs to the

definability structure. By the condition Compositionality from definition 3.9 we have that $[\![\Gamma, \Delta, \Gamma' \vdash A[B/x] : \tau]\!] = [\![\Gamma, x : \sigma, \Gamma' \vdash A : \tau]\!] \circ ([\![\Gamma]\!] \times [\![\Delta \vdash B : \sigma]\!] \times [\![\Delta]\!]$ as required.

To establish (func), suppose that $[\![\Gamma \vdash M : \sigma \to \tau]\!]$ is in R. By Concretion, $\Gamma, x : \sigma \vdash Mx : \tau$ is in the language so $[\![\Gamma, x : \sigma \vdash Mx : \tau]\!]$ is a logical form. And by the condition Concretion from definition 3.9 we see that this is $\mathbf{app} \circ ([\![\Gamma \vdash M : \sigma \to \tau]\!] \times \mathbf{id}_\sigma)$.

Establishing that that it is a Λ-definability structure is similarly straightforward. For **vac**, if $[\![\Gamma \vdash A : \sigma]\!]$ is in the structure, then $[\![\Gamma, x : \tau \vdash A : \sigma]\!] = \mathbf{vac}[\![\Gamma \vdash A : \sigma]\!]$ is also in the structure, and similarly for **id**, **merge** and **switch**. For **abs** note that if $[\![\Gamma, x : \sigma \vdash M : \tau]\!]$ is in the structure we can define $\lambda[\![\Gamma, x : \sigma \vdash M : \tau]\!] = [\![\Gamma \vdash \lambda x.M : \tau]\!]$

For 2, we must show that when $\Gamma \vdash A \in \mathcal{L}(\Lambda, \Xi, \Sigma_0)$, then $[\![\Gamma \vdash A]\!] \in R[[\![\Sigma_0]\!]]$, the Λ-minimal extension of R by $[\![\Sigma_0]\!]$. A straightforward induction shows that every line of a derivation in $\mathcal{L}(\Lambda, \Xi, \Sigma_0)$ is in $R[[\![\Sigma_0]\!]]$: derivations start with sequents of the form $\vdash c$ with $c \in \Sigma_0$, or with id if $\mathbf{id} \in \Lambda$, whose interpretations belong to $R[[\![\Sigma_0]\!]]$, and will proceed by rules that $R[[\![\Sigma_0]\!]]$ is closed under.

We must also show the converse, that $R[[\![\Sigma_0]\!]] \subseteq \{[\![\Gamma \vdash A]\!] \mid \Gamma \vdash A \in \mathcal{L}(\Lambda, \Xi, \Sigma_0)\}$. Note that $R[[\![\Sigma_0]\!]]$ is defined as the smallest type-indexed set containing R, containing $[\![\Sigma_0]\!]$, and closed under comp, fun, and the Λ-operations. So in order to show that $R \subseteq \{[\![\Gamma \vdash A]\!] \mid \Gamma \vdash A \in \mathcal{L}(\Lambda, \Xi, \Sigma_0)\}$ it suffices to show that the latter contains R, $[\![\Sigma_0]\!]$, and is closed under comp, fun, and the Λ-operations. But this is clear from the constraints on $[\![\cdot]\!]$. □

Indeed, we can use the previous result to see that every definability structure, $(D, R, \mathbf{app})$, is a structure of definable functions in *some* interpreted language, namely a language with a primitive rule for every element of R. That is to say, the interpreted language where $\Xi = R$ and with the interpretation $[\![r]\!]_R = r$.

Corollary 4.1. *Every Λ-definability structure $(D, R, \mathbf{app})$ is the structure of Λ-definable functions over $(D, \mathbf{app}, [\![\cdot]\!]_R)$ in the language $\mathcal{L}(\Lambda, R, \emptyset)$.*

...

Proposition 4.3. *Consider an interpretation $(D, \mathbf{app}, [\![\cdot]\!])$ satisfying the environment model condition for a language $\mathcal{L}(\Lambda, \Xi, \Sigma)$. Then $R[[\![\Delta]\!]] =$*

4.3 Substitution structures

The present framework for modeling propositional structure generalizes the framework of substitution structures presented in Bacon (2019). This section explains the connection. I will briefly introduce the notion of a substitution structure, and I'll show how one can given a substitutional analysis of the notion of a logical form and metaphysical definability, generalizing and improving the definitions given in Bacon (2019). Substitution structures provide us with our last class of example of definability structures; they correspond to structures for the λ-signature with all the structural rules **id**, **switch**, **merge**, **vac**.

The substitutional approach to propositional structure can be explained, again, by a linguistic analogy. In this analogy the logically simple (or "fundamental") properties and relations in reality correspond to the primitive predicates and relations of a language. Complex properties and relations can be thought of as composed of these basic properties and relations via logical forms, the metaphysical analogue of syncategorematic operations. In a language we are able to make sense of the result of substituting the simple constituents of an expression with other things of the same type as that constituent. But even if reality is not structured like a language—even if it is relatively coarse-grained—a large class of theories of propositional structure can support the metaphysical analogue of this idea of substituting simple constituents for other things: a *metaphysical substitution*.

In a given model of properties and relations (an applicative structure $(D, \mathbf{app})$), a metaphysical substitution can be modeled by a homomorphism of that applicative structure to itself: a type indexed collection of functions $i^\sigma : D^\sigma \to D^\sigma$ that commutes with application $i^\tau \mathbf{app}(d, a) = \mathbf{app}(i^{\sigma\to\tau} d, i^\sigma a)$ for every $d \in D^{\sigma\to\tau}$ and $a \in D^\sigma$, in accordance with the fact that substitutions preserve logical forms, and thus in particular preserves things in subject-predicate form. The abstract approach to substitutions allows us to make sense of substituting one thing for another in a proposition without assuming that propositions are structured enough to have explicit constituents.

Definition 4.1 (Substitution structure)**.** *A substitution structure is a triple* $(D, \mathbf{app}, M)$ *where* $(D, \mathbf{app})$ *is an applicative structure,* M *is a set of homomorphisms of* $(D, \mathbf{app})$ *to itself, called the* metaphysical substitutions*, and* M *contains the identity and is closed under composition.*

In Bacon (2019) and Bacon (2020) a restricted notion of metaphysical

definition was given in this framework which had the following meaning: $C_1, \ldots C_n$ are *logically simple* and together metaphysically define A. The analysis was based on the following way of expressing *linguistic* definition from simple expressions (constants):

> A is entirely definable using the non-logical constants $c_1, \ldots, c_n$, the logical constants and the syncategorematic rules of the language if and only if, every substitution of the language that fixes the constants $c_1 \ldots c_n$ also fixes A.

It is not too hard to see that this is equivalent to our account of linguistic definition in section 2.2—that there exists a linguistic logical form, in the sense of definition 2.4, mapping $c_1, \ldots, c_n$ to A—*provided* $c_1, \ldots, c_n$ are constants. By recasting the above definition in terms of substitution structures we obtain the notion of metaphysical definition in Bacon (2019).

However the analysis breaks down when we try to capture the relation of metaphysical definition between *arbitrary* entities $C_1, \ldots, C_n$ and B, where $C_1 \ldots C_n$ might not be metaphysically simple. Moreover, the unrestricted notion of metaphysical definition is necessary for most applications of the framework in science and philosophy. Consider, for instance, the definition of heat in terms of further thermodynamical properties like entropy, that are not themselves fundamental, or bachelorhood in terms of being married and manhood. To see why the substitutional analysis described above is not fit for this more general purpose, consider again the case of linguistic definition. We clearly do not want to say that p can be defined from the complex sentence $(p \wedge q)$. Yet any substitution that maps the latter sentence to itself, must map p to itself (as well as q and $\wedge$).

My goal here is twofold: (i) to establish a partial correspondence between the two frameworks and (ii) to supplement Bacon (2019) with a fully general substitution theoretic analysis of metaphysical definability using concepts from the present framework. To achieve (i) we will give a substitutional analysis of a logical form and, conversely, a definition of a metaphysical substitution in terms of logical form, establishing a partial correspondence between the frameworks. For (ii) we can take our substitutional analysis of logical form, and plug it into our analysis of metaphysical definability in terms of the existence of a logical form mapping the definiens to the definiendum.

The fundamental idea behind the correspondence is this: substitutions are simply mappings that preserve logical form and, conversely, logical forms are just things preserved by substitutions. Using the first idea, we can define

a logical form in terms of substitutions as something that is preserved by all substitutions. Given a substitution structure, a logical form is thus an n-ary function between the domains of the structure that commutes with all substitutions in the sense that $i\mathbf{r}(\overline{d}) = \mathbf{r}(i\overline{d})$ for any substitution i and sequence of elements $\overline{d}$ of the substitution structure.

Definition 4.2 (The logical forms of a substitution structure)**.** *Let* $S = (D, M, \mathbf{app})$ *be a substitution structure. Then we write* $\mathrm{lf}(M)$ *for the logical forms determined by* M*:*

$$(\mathrm{lf}(M))^{\sigma}_{\overline{\rho}} := \{\mathbf{r} : D^{\overline{\rho}} \to D^{\sigma} \mid \textit{for every } i \in M,\ i(\mathbf{r}(\overline{d}) = \mathbf{r}(i\overline{d})\}$$

We write $\mathrm{lf}(S) := (D, \mathrm{lf}(M), \mathbf{app})$*, which is the* $\{\mathbf{vac}, \mathbf{switch}, \mathbf{merge}, \mathbf{id}\}$*-definability structure determined by* S*:*

Two substitution structures, $(D, M, \mathbf{app})$ and $(D, M', \mathbf{app})$, based on the same applicative structure may have the same logical forms, in which case we call them *equivalent*.

Definition 4.3 (Equivalence)**.** $(D, M, \mathbf{app})$ *and* $(D, M', \mathbf{app})$ *are equivalent iff* $\mathrm{lf}(M) = \mathrm{lf}(M')$*.*

In the other direction, we define a substitution as a mapping that preserves logical form. So given a definability structure, a substitution is a homomorphism i such that $i\mathbf{r}(\overline{d}) = \mathbf{r}(i\overline{d})$ for all $\mathbf{r} \in R^{\sigma}_{\overline{\rho}}$ of the definability structure.

Definition 4.4 (The substitutions of a definability structure)**.** *Suppose that* $A = (D, R, \mathbf{app})$ *is a definability structure. Then we write* $\mathrm{sub}(R)$ *for the substitutions determined by* R*:*

$i\mathbf{r}(\overline{d}) = \mathbf{r}(i\overline{d})$ *for every* $\overline{d} \in D^{\overline{\rho}}$ *and* $\mathbf{r} \in R^{\sigma}_{\overline{\rho}}$*.*

We write $\mathrm{sub}(S)$ *for the substitution structure* $(D, \mathrm{sub}(R), \mathbf{app})$*.*

Heuristically, we can see that these definitions are correct by noting the correctness of their analogues in the case of linguistic logical forms and linguistic substitutions. The coincidence of a substitution with something that commutes with all the syncategorematic operations of a language is usually taken to be definitional of a substitution, so we will focus on the definition of a logical form in terms of substitutions. Recalling definition 2.4, a

function $\mathbf{r}$ is a linguistic logical form if and only if there exists a sequent $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash A : \tau$ involving no non-logical constants (i.e. a sequent of $\mathcal{L}(\Lambda, \Xi, \emptyset)$) such that $\mathbf{r}(C_1 \ldots, C_n) = A[\overline{C}/\overline{x}]$. Since substitutions preserve logical form — they fix the logical constants, and leave syncategorematic operations alone — substitutions clearly commute with such functions. Conversely, provided Σ^σ is non-empty at every type σ, if a function of the form $\mathbf{r} : \mathcal{L}^{\rho_1}(\Lambda, \Xi, \Sigma) \times \ldots \times \mathcal{L}^{\rho_n}(\Lambda, \Xi, \Sigma) \to \mathcal{L}^\sigma(\Lambda, \Xi, \Sigma)$ commutes with every substitution it must be of the form $\mathbf{r}(B_1 \ldots, B_n) = A[\overline{B}/\overline{x}]$ for some logical term A. For let $A[\overline{c}]$ be the sentence $\mathbf{r}(\overline{c})$ for some arbitrary sequence of constants $\overline{c}$ in Σ of appropriate types. Then by considering the substitution i mapping each c_k to B_k, we can see that $\mathbf{r}(\overline{B}) = A[\overline{B}/\overline{c}]$ for any appropriate sequence of terms $\overline{B}$: $A[\overline{B}/\overline{c}] = iA[\overline{c}] = i\mathbf{r}(\overline{c}) = \mathbf{r}(i\overline{c}) = \mathbf{r}(\overline{B})$. So $\mathbf{r}$ has the required form.

We must check that these two definition do indeed deliver us with a definability structure and a substitution structure. The latter requires just showing that $\mathrm{sub}(R)$ is a monoid:

Proposition 4.4. $\mathrm{sub}(R)$ *as defined above contains the identity homomorphism and is closed under composition.*

The proof is trivial. In the other direction we have:

Proposition 4.5. $(D, \mathrm{lf}(M), \mathbf{app})$ *is a definability structure for the* λ *signature* $\{\mathbf{id}, \mathbf{vac}, \mathbf{merge}, \mathbf{switch}\}$.

Proof. Let us write R for $\mathrm{lf}(M)$. Firstly we show that the conditions for $\mathbf{id}, \mathbf{vac}, \mathbf{merge}, \mathbf{switch}$ hold. Clearly $\mathbf{id}_\sigma \in R^\sigma_\sigma$ since the identity trivially commutes with any substitution. If $\mathbf{r} \in R^\sigma_{\overline{\rho}}$ then $i(\mathbf{switchr}(\overline{d}_1 ab\overline{d}_2) = i(r(\overline{d}_1 ba\overline{d}_2)) = \mathbf{r}(i(\overline{d}_1 ba\overline{d}_2)) = \mathbf{switchr}(i(\overline{d}_1 ab\overline{d}_2))$. Similar calculations establish the case of vac and merge.

Closure under parallel compositions is straightforward, if $\mathbf{r} \in R^\tau_{\overline{\rho}_1 \sigma \rho_3}$ and $\mathbf{p} \in R^\sigma_{\rho_2}$ then $i\mathbf{r}(\overline{d}_1 \mathbf{p}(\overline{d}_2)\overline{d}_3) = \mathbf{r}(i\overline{d}_1 i\mathbf{p}(\overline{d}_2)i\overline{d}_3) = \mathbf{r}(i\overline{d}_1 \mathbf{p}(i\overline{d}_2)i\overline{d}_3)$ so $\mathbf{app} \circ (\overline{\mathbf{id}} \times \mathbf{p} \times \overline{\mathbf{id}})$ is also substitution invariant.

Next we must show that

(func) $\mathbf{r} \in R^{\sigma \to \tau}_{\overline{\rho}}$ only if $\mathbf{app} \circ (\mathbf{r} \times \mathbf{id}) \in R^\tau_{\overline{\rho}\sigma}$

Assume that $\mathbf{r} \in R^{\sigma \to \tau}_{\overline{\rho}_1}$ and that $i \in M$. We must show that $i\mathbf{app}(\mathbf{r}(\overline{d}), e) = \mathbf{app}(\mathbf{r}(i\overline{d}), ie)$ for arbitrary $\overline{d}$ and e, which it does by the fact that i commutes with $\mathbf{app}$ (it is a homomorphism) and with $\mathbf{r}$ by assumption.

□

Next we consider the conditions under which $(D, \mathrm{lf}(M), \mathbf{app})$ is λ-closed. In general it is not, but Bacon (2019) defines a very natural class of substitution structures—quasi-full and quasi-functional structures—which turn out to always determine λ-closed definability structures:

Definition 4.5 (Quasi-full and functional substitution structures)**.** *A substitution structure is* quasi-full *iff, for every σ, τ, to every $f : M \times D^\sigma \to D^\tau$ with the property that $jf(i,a) = f(j \circ i, a)$ for all $i, j \in M, a \in D^\sigma$, we can associate an element $f^* \in D^{\sigma\to\tau}$ such that (i) $\mathbf{app}(f^*, a) = f(1, a)$ for every $a \in D^\sigma$ and (ii) $if^* = ((j,a) \mapsto f(j \circ i, a))^*$.*

It is quasi-functional iff, whenever $d, d' \in D^{\sigma\to\tau}$, if $\mathbf{app}(id, a) = \mathbf{app}(id', a)$ for every $i \in M$ and $a \in D^\sigma$, $d = d'$.

Theorem 4.1. *If $(D, M, \mathbf{app})$ is quasi-full and functional, then $(D, \mathrm{lf}(M), \mathbf{app})$ is λ-closed and so is a definability structure for the full λ signature $\{\mathbf{id}, \mathbf{vac}, \mathbf{merge}, \mathbf{switch}, \mathbf{abs}\}$.*

Proof. Suppose that $(D, \mathbf{app})$ is quasi-full and quasi-functional. Suppose $\mathbf{r} \in R^\tau_{\overline{\rho}\sigma}$. For each $\overline{d}$, the function $f_{\overline{d}}(i,a) = \mathbf{r}((i\overline{d})a)$ satisfies the condition that $jf_{\overline{d}}(i,a) = f_{\overline{d}}(j \circ i, a)$ for all $i, j \in M$, $a \in D^\sigma$. Thus by quasi-fullness there is, for each such $\overline{d}$, an element $f^*_{\overline{d}} \in D^{\sigma\to\tau}$ such that $\mathbf{app}(f^*_{\overline{d}}, a) = \mathbf{r}(\overline{d}a)$, so we can define $\lambda\mathbf{r}$ as $\overline{d} \mapsto f^*_{\overline{d}}$. Next we show that this function is in $\mathrm{lf}(M)$: that $i\lambda\mathbf{r}(\overline{d}) = \lambda\mathbf{r}(i\overline{d})$ for any $i \in M$ and appropriate sequence $\overline{d}$. This can be established using quasi-functionality: it suffices to show that $\mathbf{app}(ji\lambda\mathbf{r}(\overline{d}), a) = \mathbf{app}(j\lambda\mathbf{r}(i\overline{d}), a)$ for every $j \in M$ and $a \in D^\sigma$. The left-hand-side computes as follows:

$$\mathbf{app}(ji\lambda\mathbf{r}(\overline{d}), a) = \mathbf{app}(ji f^*_{\overline{d}}, a) = \mathbf{app}(((k,a) \mapsto f_{\overline{d}}(k \circ j \circ i, a))^*, a) = f_{\overline{d}}(j \circ i, a) = r((ji\overline{d})a)$$

and the right-hand-side simplifies to the same value:

$$\mathbf{app}(j\lambda\mathbf{r}(i\overline{d}), a) = \mathbf{app}(j f^*_{i\overline{d}}, a) = \mathbf{app}(((k,a) \mapsto f_{i\overline{d}}(k \circ j, a))^*, a) = f_{i\overline{d}}(j, a) = r((ji\overline{d})a)$$

This function is, moreover, unique. Suppose that $\mathbf{q}$ commutes with all substitutions (i.e. is in $\mathrm{lf}(M)$) and $\mathbf{app}(\mathbf{q}(\overline{d}), a) = \mathbf{r}(\overline{d}a)$ for every sequence $\overline{d}$ and a in the structure. Again we may show that $\mathbf{q} = \lambda\mathbf{r}$ using quasi-functionality. It's sufficient to prove that for arbitrary $j \in M, a \in D^\sigma$ $\mathbf{app}(j\mathbf{q}(\overline{d}), a) = \mathbf{app}(j\lambda\mathbf{r}(\overline{d}), a)$. The right-hand-side is:

$$\mathbf{app}(j\lambda\mathbf{r}(\overline{d}), a) = \mathbf{app}(j f^*_{\overline{d}}, a) = \mathbf{app}(((k,a) \mapsto f_{\overline{d}}(k \circ j, a))^*, a) = f_{\overline{d}}(j, a) = \mathbf{r}(j\overline{d}a)$$

and the left-hand-side is:

$$\mathbf{app}(j\mathbf{q}(\overline{d}), a) = \mathbf{app}(\mathbf{q}(j\overline{d}), a) = \mathbf{r}((j\overline{d})a)$$

as required. □

In order to complete our discussion of the correspondence between the frameworks, we will explain the sense in which definability structures are more general than substitution structures. Note, firstly, that we have the following.

Corollary 4.2. *Some substitution structures are not determined by any definability structure, and some definability structure are not determined by a substitution structure. I.e. neither* lf *nor* sub *are surjective.*

Proof. For the first part, consider the applicative structure of closed terms in a language $\mathcal{L}(\Lambda, \Xi, \Sigma)$ where Σ^σ is infinite at every type σ and let M be the set of substitutions of the language that map all but finitely many constants to themselves. As we showed above, $\mathbf{r} \in \mathrm{lf}(M)$ exactly when there is a closed term $A[\overline{c}]$ such that $\mathbf{r}(\overline{B}) = A[\overline{B}/\overline{c}]$ for all sequences of closed terms $\overline{B}$ with types matching $\overline{c}$. Clearly arbitrary substitutions commute such functions, so that $\mathrm{sub}\,\mathrm{lf}(M)$ contains substitutions that map an infinite number of constants to terms other than themselves.

For the second part, proposition 4.5 tells us that every substitution structure gives rise to a $\{\mathbf{id}, \mathbf{vac}, \mathbf{switch}, \mathbf{merge}\}$-definability structure, so definability structures that do not satisfy one of these conditions will not be determined by a substitution structure. □

Our example of a substitution structure that is not generated by a definability structure is rather artificial. The structure of linguistic substitutions that fix all but finitely many constants is restricted in an arbitrary way that doesn't really reflect the structure of the underlying structure of properties and relations. Earlier we said that two substitution structures on a given applicative structure are equivalent when they determine the same logical forms—equivalent structures contain the same information about the structure on the elements of different domains. So it is natural to ask whether every substitution structure is equivalent to a structure determined by a definability structure.

We can see that the answer is 'yes' as follows. From each class of substitution structures that are essentially the same, there is a structure in which the monoid M is maximal, which we will call a normal substitution structure.

Definition 4.6 (Normal substitution structure)**.** *A substitution structure* $(D, M, \mathbf{app})$ *is* normal *when* $\mathrm{sub}\,\mathrm{lf}(M) = M$*.*

Clearly all normal substitution structures are representable by a definability structure, namely $(D, \mathrm{lf}(M), \mathbf{app})$. Thus every substitution structure is essentially the same as a substitution structure that is representable by a definability structure. By contrast, a definability structure that doesn't satisfy, say, the switch condition, is not essentially the same as one that does, on any good understanding of what it might mean for two definability structures to be "essentially the same". Thus some definability structures are not representable, even up to a good notion of equivalence, by a substitution structure. In this sense, then, definability structures are more general than substitution structures.

Finally, as a straightforward corollary of this discussion, we provide, as promised, a purely substitution theoretic definition of general metaphysical definability in the framework of Bacon (2019).

Definition 4.7 (Substitutional definability)**.** *If* $(D, \mathbf{app}, M)$ *is a substitution structure,* $\overline{d} \in D^{\overline{\rho}}$*,* $a \in D^{\sigma}$*, we say that* $\overline{d}$ *substitutionally defines* a *iff there exists a function* $\mathbf{r} : D^{\overline{\rho}} \to D^{\sigma}$ *such that*

(i) For all $i \in M$*,* $\overline{e} \in D^{\overline{\rho}}$*,* $i\mathbf{r}(\overline{e}) = \mathbf{r}(i\overline{e})$

(ii) $\mathbf{r}(\overline{d}) = a$

To see that the analysis of metaphysical definition really generalizes that given in Bacon (2019) we must show that they coincide when the definiens are fundamental. Here are two observations:

Proposition 4.6. *If* $\overline{d}$ *substitutionally defines* a *(in a substitution structure) then every substitution that fixes* $\overline{d}$ *fixes* a*.*

Proof. Suppose $\mathbf{r}(\overline{d}) = a$ where $\mathbf{r}$ is substitution invariant, and $i\overline{d} = \overline{d}$. Then $ia = i\mathbf{r}(\overline{d}) = \mathbf{r}(i\overline{d}) = \mathbf{r}(\overline{d}) = a$. □

Conversely, if $\overline{d}$ are elements of a fundamental basis, in the sense of Bacon (2019) definition 19, then we can obtain the converse: if every substitution that fixes each element of $\overline{d}$ fixes a, then there exists a substitution invariant $\mathbf{r}$ such that $\mathbf{r}(\overline{d}) = a$, by setting $\mathbf{r}(\overline{b}) = ia$ where i is a substitution extending a mapping i^{-} such that $i^{-}(\overline{d}) = \overline{b}$ (i.e. $i^{-}(d_k) = b_k$ for each $k = 1, \ldots, n$).[21]

[21]That $\mathbf{r}$ commutes with substitions can be established as follows. $j\mathbf{r}(\overline{b}) = ia$ where

5 Logics of Metaphysical Definability

In this section we axiomatize the logics of logical definability over various classes of definability structures, as well as presenting some partial results concerning the the logic of conditional definability. The goal is to come up with an axiomatization of these notions that is sound and complete with respect to different classes of models.

We will axiomatize the logics of Λ-definability models in the case that $\mathbf{id} \in \Lambda$, since things run fairly smoothly in these cases. In section 5.5 we discuss the complications that arise with extending the results to the cases where $\mathbf{id} \notin \Lambda$.

5.1 Logics of definability

Throughout this section we will work in the language $\mathcal{L}^+(\Lambda, \Xi, \Sigma)$ containing primitives for expressing logical definability statements. Recall that in the case that $\mathbf{id} \in \Lambda$ we add $\rhd$ by the following type of rule

$$\frac{\Gamma_1 \vdash A_1 : \sigma_1 \qquad \ldots \qquad \Gamma_n \vdash A_n : \sigma_n \qquad \Delta \vdash B : \tau}{\Gamma_1 \ldots \Gamma_n \Delta \setminus \overline{x} \vdash A_1 \ldots A_n \rhd^{\overline{\sigma}\tau}_{\overline{x}} B}$$

$\Theta \setminus \overline{x}$ is the result of removing all occurences of the variables $x_1, \ldots, x_n$ from the context Θ. $\rhd_{x_1 \ldots x_n}$ is shorthand for $\rhd_{\{x_1, \ldots, x_n\}}$ so that $\rhd$ is really indexed by finite sets of variables, not sequences, and $\rhd_{xy}$ and $\rhd_{yx}$ are the same expression.

This language suffices for the purposes of introducing deductive systems axiomatizing the class of λ-closed structures. In order to construct an adequate deductive system for Λ-structures, we will also need to impose the condition that Ξ has infinitely many syncategorematic rules of the form $r : \overline{\rho} \Rightarrow \tau$ for every sequence of types $\overline{\rho}\tau$. Their purpose is akin to the role that variables play in deductive systems for quantifiers (where a proof that an arbitrary object, represented by a variable, has some condition suffices for one to infer the universal generalization). We will therefore call these extra elements of Ξ *rule variables*. Variables are often treated as technical devices in proof systems and are not regarded as interpreted expressions. Usually

$i\overline{d} = \overline{b}$. And $\mathbf{r}(j\overline{b}) = ka$ where $k\overline{d} = j\overline{b} = ij\overline{d}$. Now by proposition 8 of Bacon (2019), it follows that since $k\overline{d} = ji\overline{d}$, and since every substitution that fixes $\overline{d}$, fixes a, that $ka = ija$, so that $j\mathbf{r}(\overline{b}) = \mathbf{r}(j\overline{b})$ as required.

a more complicated proof system is available in which these uninterpreted expressions do not occur, but I will not engage seriously with the project of eliminating variables of either sort here. When **abs** $\in \Lambda$, however, deductive rule variables *are* eliminable in a straightforward way which will be explained later.

We assume some basic logical axioms and rules. The system HF consists of the smallest subset of the type t terms of $\mathcal{L}^+(\Lambda, \Xi, \Sigma)$ in some signature of non-logical constants Σ containing all instances of the following principles and closed under the rules MP and Gen.

PC All instances of propositional tautologies.

UI $\forall_\sigma x.A \to A[t/x]$ where t is a term free for x in A

$\beta\eta$ $A \to A'$ when A and A' are immediate β or η equivalents.

Refl $A =_\sigma A$

LL $A =_\sigma B \to C \to C[B/A]$ provided no free variables in A or B are in the scope of variable binders in C.

Formal Functionality $\forall \overline{x}(A =_\sigma B) \to C \to C[A/B]$ provided any free variables in A or B that are in the scope of variable binders in C belong to $\overline{x}$.

MP If $\vdash A \to B$ and $\vdash A$ then $\vdash B$.

Gen If $\vdash A \to B$ then $\vdash A \to \forall_\sigma x.A$ when $x \notin FV(A)$

By HF^Λ we will mean all instances of these axioms in the language $\mathcal{L}^+(\Lambda, \Xi, \Sigma)$. Observe that these schemas above, and throughout the paper, are not only schematic in the expressions that can take the place of the latin letters (like A, B, t above), but they are also schematic in the types of lower case Greek letters (e.g. σ above).

Formal Functionality is a result of our simplifying assumption of modeling propositional functions in terms of set-theoretic functions. This partially captures the idea that propositional functions are individuated by their outputs.[22] In languages that contain λ-abstracts (i.e. when **abs** $\in \Lambda$)

[22]It is only a partial characterization since it only applies to propositional functions determined by sequents $\overline{x} : \overline{\sigma} \vdash A$, expressible in of the language $\mathcal{L}(\Lambda, \Xi, \Sigma)$.

and $\beta\eta$ Formal Functionality is equivalent to the principle Functionality $\forall_{\sigma\to\tau}FG(\forall_\sigma x(Fx =_\tau Gx) \to F =_{\sigma\to\tau} G)$, described in, e.g., Bacon (2023b) §6.5.[23] When λ is missing from the language the principles are inequivalent.

For each choice of λ-signature Λ, one can ask what the logic of logical definability is in $\mathcal{L}^+(\Lambda, \Xi, \Sigma)$ over the class of Λ-definability structures, and one can similarly ask what the logic of conditional definability is over these structures. In the remainder we will axiomatize 64 logics out of the 128 possible logics in the first class, and 32 from the latter.

Definition 5.1 (Logics of definability)**.**

- $\mathbf{L}[\Lambda]$ *is the set of sentences in the language* $\mathcal{L}^+(\Lambda, \Xi, \Sigma)$ *of logical definability true in all* Λ*-definability structures.*
- $\mathbf{L}^c[\Lambda]$ *is the set of sentences of the language of conditional definability* $\mathcal{L}^+(\Lambda, \Xi, \Sigma)$ *that are true in all* Λ*-definability structures.*

Note that our logics are parameterized by the λ-signature Λ, which determines which sentences belong to the logic $\mathbf{L}(\Lambda)$ in two distinct ways. First, Λ determines which language the sentences belong to, and in particular, which λ-terms can appear in those sentences. Second, Λ also determines which class of definability structures are relevant (Λ-structures), and so which sentences are valid may vary in a way that outstrips varying which formulas well-formed in the language.

5.2 The logic of logical definability

Henceforth we will use Greek letters like Γ, Δ and so on as short for sequences of terms of the right type to make $\Gamma \rhd \frac{\overline{\sigma}}{\overline{x}} M$ well formed. The logic of logical definability $\mathbf{L}[\emptyset]$ must contain the following principles, which are valid in *all* definability structures. We call the result of adding these principles to H the *base principles*. They are summarized in table 5, but must be read as follows. In the principle Instantiation $\overline{z}$ are free in B and $\overline{xz}$ must not get bound when B is substituted for y in A or any term in Γ. In Rules $r : \overline{\theta} \Rightarrow \sigma \in \Xi$. In Cut

[23]Formal Functionality allows us to move from $\forall_\sigma x(Fx =_\tau Gx)$ to $\lambda x.Fx = \lambda x.Fx \to \lambda x.Fx = \lambda x.Gx$, and using Refl and η we can conclude $F = G$. Conversely, if $\forall\overline{x}(A = B)$ we can conclude using β that $\forall\overline{x}((\lambda\overline{x}.A)\overline{x} = (\lambda\overline{x}.B)\overline{x})$. By Functionality $\lambda\overline{x}.A = \lambda\overline{x}.B$. C is β equivalent to $C[(\lambda\overline{x}.B)\overline{x}/B]$, which by Leibniz's law is $C[(\lambda\overline{x}.A)\overline{x}/B]$ which is β equivalent to $C[A/B]$ as required.

Instantiation	$\Gamma \rhd^{\overline{\sigma}\rho}_{\overline{x}y} A \to \Gamma[B/y] \rhd^{\overline{\sigma}\rho}_{\overline{x}z} A[B/y]$
Rules	$\Gamma \rhd^{\overline{\theta}\sigma}_{\overline{x}} r(\Gamma)$
Cut	$(\Gamma \rhd^{\overline{\theta}\sigma}_{\overline{x}} A) \wedge (\Delta, A, \Delta' \rhd^{\overline{\rho}_1\sigma\overline{\rho}_2}_{\overline{y}} B) \to (\Delta, \Gamma, \Delta' \rhd^{\overline{\rho_1\theta\rho_2}}_{\overline{y}} B)$
Concretion	$\Gamma \rhd^{\overline{\rho}(\sigma\to\tau)}_{\overline{x}} F \to \Gamma, y \rhd^{\overline{\rho}\sigma\tau}_{\overline{x}y} Fy$
Rule Gen	If $\vdash C \to \exists\overline{x}.r(\overline{A}) \neq B$ then $\vdash C \to \neg(\overline{A} \rhd_{\overline{x}} B)$

Table 5: The Base Principles

Id	$x \rhd^{\sigma}_{\overline{y}} x$
Switch	$\Gamma, x, y, \Delta \rhd_{\overline{z}} C \to \Gamma, y, x, \Delta \rhd_{\overline{z}} C$
Merge	$\Gamma, y, \Delta, y, \Sigma \rhd_{\overline{x}} C \to \Gamma, y, \Delta, \Sigma \rhd_{\overline{x}} C$
Vac	$\Gamma \rhd_{\overline{x}} C \to \Gamma, y \rhd_{\overline{x}y} C.$
Abs	$\Gamma, y \rhd^{\overline{\rho}\sigma\tau}_{\overline{x}y} My \to \Gamma \rhd^{\overline{\rho}(\sigma\to\tau)}_{\overline{x}} M$

Table 6: The Λ Principles

we require that $\{\overline{x}\} \subseteq \{\overline{y}\}$, in Concretion $y \in \overline{x} \setminus FV(\Gamma, F)$ and in Rule Gen r is a rule variable that doesn't appear in C.

What principles characterize Λ-definability structures for $\Lambda \subseteq \{\mathbf{id}, \mathbf{switch}, \mathbf{merge}, \mathbf{vac}, \mathbf{abs}\}$? We call these the Λ principles; these are summarized in table 6. We say that a variable is fresh in $\Gamma \rhd C$ when it doesn't appear free in Γ or C. The variables x and y to be fresh in all of the schemas in table 6 and in Abs we require $y \in \{\overline{x}\} \setminus FV(\Gamma, M)$.

The rule Rule Gen preserves validity in all definability structures. Moreover, each of the principles Id, Switch, Merge, Vac, and Abs are valid in Λ-definability structures for which Λ contains the correspondingly named conditions — i.e. they are valid in definability structures satisfying the corresponding condition id, switch, merge, vac and abs (being λ-closed). Id ensures that every thing can be defined from itself, via the vacuous Russellian function **id**, and so is valid in **id**-definability structures. Switch says says if you can define C from some things then you can also define C from those same things in a different order, and so is valid in **switch**-definability structures, whose Russellian functions are closed under switch. And so on.

The situation with logics containing Abs (when $\mathbf{abs} \in \Lambda$) is special. By using Rule Gen and Abs we can derive the following principle:

Decomposition $\Gamma, A \rhd^{\overline{\rho}\sigma}_{\overline{x}} B \to \exists_{\overline{\rho}\to\sigma} F(\Gamma \rhd^{\overline{\rho}\to\sigma}_{\emptyset} F \wedge \forall\overline{x}.FA =_{\sigma} B)$

Bacon (2020) uses Decomposition as the basis of a definition of metaphysical

definability.[24]

This implication actually goes in the opposite direction as well. If we remove Rule Gen from our system, but we have both Abs and Decomposition, it turns out that Rule Gen is admissible. This is significant: Rule Gen is a somewhat unfamiliar kind of rule, since it involves adding "syncategorematic rule variables" to our language. Aside from this, I think it is generally more perspicuous to eliminate primitive rules in favour of axioms where it is possible. It follows that in cases where **abs** $\in \Lambda$ we can axiomatize our logics without Rule Gen and rule variables, by adding Decomposition and principles corresponding to the elements of Λ. Proposition 5.1 below summarizes the situation.

Let Λ^* be the schemas capitalized above corresponding to the elements of Λ (so that Λ^* contains the schema Merge iff Λ contains **merge**, and so on, and both Decomposition and Abstraction correspond to **abs**).

Definition 5.2 (Deductive systems)**.** *We write $\vdash_\Lambda$ for the deductive system corresponding to the λ-signature Λ. It is the smallest subset of $\mathcal{L}(\Lambda, \Xi, \Sigma)$ satisfying the following, where we write $\vdash_\Lambda A$ for $A \in \vdash_\Lambda$*

- $\vdash_\Lambda A$ *whenever A is an instance of an axiom of $\mathsf{HF}^{\Lambda\Xi\Sigma}$, a base principle or of Λ^*.*
- *If $\vdash_\Lambda A \to B$ and $\vdash_\Lambda A$ then $\vdash_\Lambda B$*
- *If $\vdash_\Lambda A \to B$ and x is a variable not appearing free in A, $\vdash_\Lambda A \to \forall_\sigma x.A$.*
- *If $\vdash_\Lambda A \to \exists \overline{x}.r(\overline{B}) \neq C$ and r doesn't appear in A, then $\vdash_\Lambda A \to \neg(\overline{B} \rhd_{\overline{x}} C)$.*

Definition 5.3 (Provability and consistency)**.** *When $T \subseteq \mathcal{L}(\Lambda, \Xi, \Sigma)$ is a set of formulas,*

- *$T \vdash_\Lambda B$ if and only if, for some finite set of formulas $A_1, \ldots, A_n \in T$, $\vdash_\Lambda (A_1 \wedge \ldots \wedge A_n) \to B$.*
- *$T \vdash_\Lambda$, or "T is Λ-inconsistent", if and only if $T \vdash_\Lambda C$ for every formula C of $\mathcal{L}(\Lambda, \Xi, \Sigma)$.*

[24] In terms of "purity", special case of definability statements $\Gamma \rhd_{\overline{x}} A$ were $\Gamma = \epsilon$.

It is immediate from the definitions and propositional logic that the deduction theorem holds: If $T, A \vdash_\Lambda B$ then $T \vdash_\Lambda A \to B$, and that $T, \neg A \vdash_\Lambda$ if and only if $T \vdash_\Lambda A$.

First we show how to eliminate Rule Gen in the case that $\mathbf{abs} \in \Lambda$.

Proposition 5.1. *Suppose* $\mathbf{abs} \in \Lambda$ *and consider the variant system* $\vdash^*_\Lambda$ *obtained from our definition of* $\vdash_\Lambda$ *by removing Rule Gen and adding Decomposition as an axiom .*

1. *Rule Gen is admissible for* $\vdash^*_\Lambda$.
2. *Decomposition is a theorem of* $\vdash_\Lambda$.

Thus $\vdash^*_\Lambda$ *and* $\vdash_\Lambda$ *are the same relation.*

Proof. For 1 suppose that $\vdash_\Lambda A \to \exists \overline{x}.r(\overline{B}) \neq C$. The contrapositive of the $\Gamma = \epsilon$ instance of Decompositions says: $\forall_{\overline{\rho} \to \sigma} F(\rhd^{\overline{\rho} \to \sigma}_\emptyset F \to \exists \overline{x}.F\overline{B} \neq_\sigma C) \to \neg B \rhd_{\overline{x}} C$. Let $F : \overline{\rho} \to \sigma$ be an arbitrary entity such that $\rhd_\emptyset F$. By replacing the rule $r(\overline{B})$ with $F\overline{B}$ everywhere in the derivation of $\vdash_\Lambda A \to \exists \overline{x}.r(\overline{B}) \neq C$, we can see that $\vdash_\Lambda A \to \exists \overline{x}.F(\overline{B}) \neq C$. Using the contrapositive of Decomposition and our assumption $\rhd_\emptyset F$ we can infer, appealing to the deduction theorem, that $\vdash_\Lambda A \to \neg(\overline{B} \rhd_{\overline{x}} C)$ as required.

For 2 we derive the contrapositive. First note that we can prove $\forall_{\overline{\rho} \to \sigma} F(\Gamma \rhd^{\overline{\rho} \to \sigma}_\emptyset F \to \exists \overline{x}.FA \neq_\sigma B) \to \exists \overline{x}.p(\Gamma A) \neq_\sigma B$, where p is a rule variable. This is established as follows. First, an instance of UI is $\forall_{\overline{\rho} \to \sigma} F(\Gamma \rhd^{\overline{\rho} \to \sigma}_\emptyset F \to \exists \overline{x}.FA \neq_\sigma B) \to (\Gamma \rhd^{\overline{\rho} \to \sigma}_\emptyset \lambda y.p(\Gamma y) \to \exists \overline{x}.\lambda y.p(\Gamma y)A \neq_\sigma B)$ instantiating F with $\lambda y.p(\Gamma y)$. We can also establish $\Gamma \rhd^{\overline{\rho} \to \sigma}_\emptyset \lambda y.p(\Gamma y)$ since $\Gamma, y \rhd_y p(\Gamma y)$ from Rules, and then by applying β and Abstraction. Note that this requires us to be able abstract into the scope of a rule variable p. By Rule Gen, we can infer $\forall_{\overline{\rho} \to \sigma} F(\Gamma \rhd^{\overline{\rho} \to \sigma}_\emptyset F \to \exists \overline{x}.FA \neq_\sigma B) \to \neg(\Gamma, A \rhd_{\overline{x}} B)$, which is equivalent to the contrapositive of Decomposition. □

In order to get a feel for these axioms, we now derive a couple of theorems in these systems.

Proposition 5.2. *Complication follows from the base principles, and Uniformity from the base principles and Abs.*

Complication $\Gamma \rhd^{\overline{\rho}_1 (\sigma \to \tau)}_{\overline{x}} M \wedge \Delta \rhd^{\overline{\rho}_2 \sigma}_{\overline{y}} \rhd N \to \Gamma, \Delta \rhd^{\overline{\rho_1 \rho_2} \tau}_{\overline{xy}} MN$

Uniformity $\Gamma \rhd^{\overline{\rho} \sigma}_\emptyset A \to \forall_{\overline{\rho}} \overline{X} \exists_\sigma Y (\overline{X} \rhd_\emptyset Y)$

Proof. Assume $\Gamma \rhd_{\overline{x}} M$ and $\Delta \rhd_{\overline{y}} \rhd N$. From Concretion we get $\Gamma, z \rhd_{\overline{x}z} Mz$, and so by Instantiation we get $\Gamma, N \rhd_{\overline{xy}} MN$. Finally we obtain the result, $\Gamma, \Delta \rhd_{\overline{xy}} MN$ by Cut.

Suppose now $\overline{B} \rhd_{\emptyset} A$. By Decomposition, $\exists F(\rhd F \wedge F\overline{B} = A)$. From $\rhd F$ we can infer from Concretion and Instantiation $X_1 \rhd FX_1$, and by repeating this we can get $\overline{X} \rhd F\overline{X}$. By existential generalization we get $\exists Y.(\overline{X} \rhd Y)$ and the result follows by universal generalization and the deduction theorem. □

Let me end by mentioning a class of principles we haven't discussed yet but are tempting principles, stating that logical operations, or even the definability operations themselves, can be used in logical metaphysical definitions: for instance, that *being old and wise* can be defined from *being old* and *being wise* alone — conjunction isn't needed as a further ingredient, it is rather a mode of combination like application and λ-abstraction are.

Conditionals $\Gamma \rhd_{\overline{x}} A \wedge \Delta \rhd_{\overline{y}} B \rightarrow \Gamma\Delta \rhd_{\overline{xy}} (A \rightarrow B)$

Equality $\Gamma \rhd_{\overline{x}} A \wedge \Delta \rhd_{\overline{y}} B \rightarrow \Gamma\Delta \rhd_{\overline{xy}} (A =_{\sigma} B)$

Universals $\Gamma \rhd_{\overline{xy}} A \rightarrow \Gamma \rhd_{\overline{x}} \forall_{\sigma} y.A$

Definitions $\Gamma_1 \rhd_{\overline{x}_1} A_1 \wedge \ldots \Gamma_n \rhd_{\overline{x}_n} A_n \wedge \Delta \rhd_{\overline{y}} B \rightarrow \overline{\Gamma}\Delta \rhd_{\overline{\overline{xy}\setminus\overline{z}}} (\overline{A} \rhd_{\overline{z}} B)$

These are validated by requiring that $[\![\Gamma \vdash A : \sigma]\!] \in R^{\sigma}_{\rho}$ for all terms A in the signature containing the logic constants. For universals, for instance, this is equivalent to stipulating that the operations **uni** from section 3.5 map elements of R to elements of R. Generalizing the soundness and completeness results of the next section to logics containing these principles, with respect to the obvious classes of definability structures, is utterly straightforward.

Because we have not built into our framework the assumption that the logical operations $\rightarrow, \forall, =$ and so on can appear in logical forms, we will single out a special language whose sequents *always* denote logical forms in the semantics (see 3.10). Let $\mathcal{L}^{-}(\Lambda, \Xi, \Sigma)$ be the system obtained by removing uni, cond and eq from the system characterizing $\mathcal{L}(\Lambda, \Xi, \Sigma)$.

Lemma 5.1. *For any sequent $\overline{x} : \overline{\sigma} \vdash C : \tau$ of $\mathcal{L}^{-}(\Lambda, \Xi, \emptyset)$, $\vdash_{\Lambda} \overline{x} \rhd_{\overline{x}} C$.*

Proof. This proceeds by an induction on proofs of the sequents of $\mathcal{L}(\Lambda, \Xi, \Sigma)$. We do the case where $\Lambda = \{\mathbf{id}, \mathbf{switch}, \mathbf{merge}, \mathbf{vac}, \mathbf{abs}\}$; for smaller λ-signatures some of the cases needed in this case are unnecessary

id. $x : \sigma \vdash x : \sigma$, therefore $x \rhd_x x$ since Id Λ^*

Rules. Suppose that $\overline{x}_1 \rhd_{\overline{x}_1} M_1, \ldots, \overline{x}_n \rhd_{\overline{x}_1} M_n$ are derivable. The axiom Rules says that $M_1, \ldots, M_n \vdash_{\overline{x}_1 \ldots \overline{x}_n} r(M_1 \ldots M_n)$. The desired ruselt, $\overline{x}_1 \ldots \overline{x}_n \rhd_{\overline{x}_1 \ldots \overline{x}_n} r(M_1 \ldots M_n)$ follows by Cut.

Switch. Suppose that $\overline{x}yz\overline{w} \vdash_{\overline{x}yz\overline{w}} M$. Then $\overline{x}zy\overline{w} \vdash_{\overline{x}yz\overline{w}} M$ follows by Switch.

Merge. Suppose $\overline{w}_1 x \overline{w}_2 y \overline{w}_3 \rhd_{\overline{w}_1 x \overline{w}_2 y \overline{w}_3} M$. Since $\overline{w}_1 x \overline{w}_2 y \overline{w}_3$ are all distinct (because they correspond to a type assignment), we can infer by Instantiation that $\overline{w}_1 z \overline{w}_2 z \overline{w}_3 \vdash_{\overline{w}_1 z \overline{w}_2 \overline{w}_3} M[z/xy]$ Then by merge $\overline{w}_1 z \overline{w}_2 \overline{w}_3 \vdash_{\overline{w}_1 z \overline{w}_2 \overline{w}_3} M[z/xy]$

Vac. Suppose $\overline{x} \rhd_{\overline{x}} M$. Then $\overline{x}y \rhd_{\overline{x}y} M$ follows immediately by the axiom Vac.

Abs. Suppose $\overline{x}y \rhd_{\overline{x}y} M$. By β this is equivalent to $\overline{x}y \rhd_{\overline{x}y} (\lambda y.M)y$. So by Abstraction $\overline{x} \rhd_{\overline{x}} (\lambda y.M)$ follows.

Constants does not occur because we've set $\Sigma = \emptyset$. □

It is clear that if we add the axioms Universal, Equality, and Conditionals to the base and Λ principles, the above result extends to all sequents of $\mathcal{L}(\Lambda, \Xi, \emptyset)$, and if we further add Definitions it extends to $\mathcal{L}^+(\Lambda, \Xi, \emptyset)$.

5.3 Soundness and Completeness

The goal of this section is to establish soundness and completeness for the logics of logical definability.

Theorem 5.1 (Soundness and Completeness)**.** *For each λ-signature Λ containing* **id***:*

$$A \in \mathbf{L}[\Lambda] \text{ iff } \vdash_\Lambda A.$$

Given a model $M = (D, R, \mathbf{app}, [\![\cdot]\!], v)$ of a language $\mathcal{L}(\Lambda, \Xi, \Sigma)$ with a rule signature Ξ that does not contain infinitely many rules at all types (i.e. doesn't contain rule variables), we can introduce the notion of *validity* of a formula A that *does* contain rule variables in this model. Suppose $\Xi^+ \supseteq \Xi$ is the extension of Ξ with infinitely many rule variables, and similarly $\Sigma^+ \supseteq \Sigma$.

Definition 5.4 (Validity)**.** *Let $M = (D, R, \mathbf{app}, [\![\cdot]\!], v)$ be a model for $\mathcal{L}(\Lambda, \Xi, \Sigma)$, and A a sentence of $\mathcal{L}(\Lambda, \Xi^+, \Sigma^+)$. We will say that $M \models A$ iff for all interpretation functions $[\![\cdot]\!]^+ : \Xi^+ \to R$, $[\![\cdot]\!]^+ : \Sigma^+ :\to D$ extending $[\![\cdot]\!]$ to Ξ^+ and Σ^+, $(D, R, \mathbf{app}, [\![\cdot]\!]^+, v) \vdash A$*

A formula A is valid when it remains true no matter how the rule variables are interpreted as logical forms (recall that by the definition of an interpretation $[\![r]\!]^+ \in R$ for any rule r). It is clear that if $A \in \mathcal{L}(\Lambda, \Xi, \Sigma)$ and is valid then $A \in \mathbf{L}(\Delta)$.

Proposition 5.3 (Soundness)**.** *The following hold with respect to a language with infinitely many rule variables $p : \overline{\rho} \Rightarrow \tau$ for each sequence $\overline{\rho}\tau$.*

1. *The logical axioms are valid, and Gen preserves validity.*
2. *The Base Principles (Instantiation, Rules, Cut, and Concretion) are valid in all definability models.*
3. *The Λ Principles are valid in all Λ-definability models.*
4. *Rule Gen preserves validity in all definability models.*

Proof. The proof of the validity of the logical axioms is standard (see e.g. Bacon (2023b) §15.2), except for Formal Functionality, and $\beta\eta$. We begin with the former. Suppose that $v([\![\forall \overline{x}(A = B)]\!]^g) = 1$ where $[\![\cdot]\!]$ is an interpretation of the signatures Ξ^+, Σ^+. Since $\overline{x}$ include all the free variables in A and in B that get bound in C, $[\![A]\!]^{g'} = [\![B]\!]^{g'}$ for all assignments $g'[\overline{x}]g$, so we may apply Compositionality$_1$ to conclude that $[\![C]\!]^g = [\![C[A/B]]\!]^g$.

For η we must show that $\lambda x.Mx$ is intersubstitutable with M, provided x is not free in M. By Compositionality, it suffices to show that $[\![\Gamma \vdash \lambda x.Mx : \sigma \to \tau]\!] = [\![\Gamma \vdash M : \sigma \to \tau]\!]$. Note that our definition of $\lambda[\![\Gamma, x : \sigma \vdash Mx : \tau]\!]$ is the unique function $\mathbf{p} : D^{\overline{\rho}\sigma} \to D^{\tau}$ (not necessarily in R) such that the following equation holds.

$$\mathbf{app} \circ (\mathbf{p} \times \mathbf{id}) = [\![\Gamma, x : \sigma \vdash Mx : \tau]\!]$$

However, Concretion says that this equation holds when $\mathbf{p} = [\![\Gamma \vdash M]\!]$. It follows that $[\![\Gamma \vdash M]\!] = \lambda[\![\Gamma, x : \sigma \vdash Mx]\!]$ by the uniqueness of $\mathbf{p}$. Finally the right-hand-side is $[\![\Gamma \vdash \lambda x.Mx]\!]$ by the condition for abstraction in definition 3.9.

□

Theorem 5.2 (Completeness)**.** *If A is valid in every Λ-definability model then $\vdash_\Lambda A$.*

Proof. Suppose that T is a Λ consistent theory, i.e. $T \nvdash_\Lambda$. We can consider T as formulated in the language $\mathcal{L}^+(\Lambda, \Xi, \Sigma)$ where Σ contains infinitely many unused constants at each type, σ, and Ξ contains infinitely many unused "rule variables" $r : \overline{\rho} \Rightarrow \sigma$ — i.e. constants and rules that make no appearance in T. By a standard procedure (see e.g. Bacon (2023b) §15.3) we can extend this into a negation complete, witness complete theory T'. This means that for any formula A, $A \in T'$ or $\neg A \in T'$, and if $\exists x.A \in T'$ then there is some constant c such that $A[c/x] \in T'$.

Next we extend T' to a "rule witness complete" theory T'', which means that for any definability statement $\overline{A} \rhd_{\overline{x}} B$, either $\neg(\overline{A} \rhd_{\overline{x}} B) \in T''$ or $\forall \overline{x}(r(\overline{A}) = B) \in T''$ for some rule variable r. First we enumerate all the variable rules r_n, and all closed formulas C_n of the new language of the form $\overline{A} \rhd_{\overline{x}} B$ in such a way that r_n only ever appears in formulas C_k with $k > n$. The procedure is built up in stages as follows with $T_0 = T'$ and

$$T_{n+1} = \begin{cases} T_n \cup \{\forall \overline{x}(r_n(\overline{A}) = B\} & \text{if this result is } \vdash_\Lambda \text{ consistent and } C_n = \overline{A} \rhd_{\overline{x}} B \\ T_n \cup \{\neg \overline{A} \rhd_{\overline{x}} B\} & \text{if not, and } C_n = \overline{A} \rhd_{\overline{x}} B \end{cases}$$

Observe that if T_n is consistent, so is T_{n+1}. For if $T_n \cup \{\forall \overline{x}.r_n(\overline{A}) = B\}$ is inconsistent, then by the deduction theorem, $T_n \vdash_\Lambda \exists \overline{x}.r_n(\overline{A}) \neq B$, and so $T_n \vdash \neg \overline{A} \rhd_{\overline{x}} B$ since r_n does not appear in T_n. So $T_n \cup \{\neg \overline{A} \rhd_{\overline{x}} B\}$ must be consistent if T_n is. Thus T_{n+1} is consistent.

We turn our theory T'' into a definability model as follows. For a closed term M of type σ we write $[M]$ for $\{N : \sigma \mid M =_\sigma N \in T''\}$. We will write $[\overline{M}]$ for a sequence $[M_1], \ldots, [M_n]$ and we will write $[\overline{A}] \rhd [B]$ as shorthand for $\overline{A} \rhd B \in T$.

- $D^\sigma := \{[M] \mid\vdash M : \sigma \text{ is a closed term of } \mathcal{L}(\Lambda, \Xi^+, \Sigma^+)\}$
- $[\![\overline{x} : \overline{\rho} \vdash C : \tau]\!] := ([A_1], \ldots, [A_n]) \mapsto [C[\overline{A}/\overline{x}]]$. $[\![c]\!] = [c]$ for $c \in \Sigma$, $[\![r]\!] = [\overline{M}] \mapsto [r(\overline{M})]$ for $r \in \Xi$.
- $R^\sigma_{\overline{\rho}} := \{[\![\overline{x} : \overline{\rho} \vdash C : \sigma]\!] : D^{\overline{\rho}} \to D^\sigma \mid \overline{x} : \overline{\rho} \vdash C : \sigma \in \mathcal{L}^-(\Lambda, \Xi^+, \emptyset)\}$.
- $\mathbf{app}([M], [N]) = [MN]$,
- $v([A]) = 1$ iff $A \in T$.

Note that logical forms in this model are determined by sequents in the language without logical constants, *or* the syncategorematic operations $\to$

$, \forall, =$. The reason we do not include the latter operations in our logical forms is because we are remaining neutral about whether they are logical. We will write $[\overline{M}]$ for a sequence $[M_1], \ldots, [M_n]$.

First we need to show that $(D, R, \mathbf{app})$ is a Λ-definability structure, then we need to show that $[\![\cdot]\!]$ satisfies the environment model condition on it, and finally we need to show it is a definability model by showing that v satisfies the relevant truth clauses.

To be a definability structure we must establish (comp) and (func). For (comp) suppose that $\mathbf{r} = [\![\overline{x} : \overline{\theta}_1, y : \sigma, \overline{z} : \overline{\theta}_2 \vdash A]\!] \in R^{\tau}_{\overline{\theta}_1 \sigma \overline{\theta}_n}$, and $\mathbf{p} = [\![\overline{w} : \overline{\rho} \vdash B : \sigma]\!] \in R^{\sigma}_{\overline{\rho}}$. We want to show that $\mathbf{r} \circ (\overline{\mathbf{id}}_{\overline{\theta}_1} \times \mathbf{p} \times \overline{\mathbf{id}}_{\overline{\theta}_2}) \in R^{\tau}_{\overline{\theta_1 \rho \theta_1}}$. The substitution lemma guarantees that the sequent $\overline{x} : \overline{\theta}_1, \overline{w} : \overline{\rho}, \overline{z} : \overline{\theta}_2 \vdash B[A/x] : \sigma$ is derivable, and the function $[\![\overline{x} : \overline{\theta}_1, \overline{w} : \overline{\rho}, \overline{z} : \overline{\theta}_2 \vdash B[A/x] : \sigma]\!]$ does the job. That $\mathbf{r} \circ (\overline{\mathbf{id}}_{\overline{\theta}_1} \times \mathbf{p} \times \overline{\mathbf{id}}_{\overline{\theta}_2}) = [\![\overline{x} : \overline{\theta}_1, \overline{w} : \overline{\rho}, \overline{z} : \overline{\theta}_2 \vdash B[A/x] : \sigma]\!]$ is exactly the statement of the Compositionality$_2$ established below.

For (func) suppose that $\mathbf{r} = [\![\overline{x} : \overline{\rho} \vdash F : \sigma \to \tau]\!] \in R^{\sigma \to \tau}_{\overline{\rho}}$. We want to show that $\mathbf{app} \circ (\mathbf{r} \times \mathbf{id}_{\sigma}) \in R^{\tau}_{\overline{\rho} \sigma}$. By concretion the sequent $\overline{x} : \overline{\rho}, y : \sigma \vdash Fy : \tau$ is in $\mathcal{L}^{-}(\Lambda, \Xi, \emptyset)$. Moreover $[\![\overline{x} : \overline{\rho}, y : \sigma \vdash Fy : \tau]\!] = \mathbf{app} \circ (\mathbf{r} \times \mathbf{id}_{\sigma})$ because this is the statement of the semantic clause for concretion, which is established below.

Next we show that this definability structure is a Λ-definability structure. Clearly, $\mathbf{id} = [\![x : \sigma \vdash x : \sigma]\!] \in R^{\sigma}_{\sigma}$. If $\mathbf{switch} \in \Lambda$ and $[\![\overline{z} : \overline{\rho}_1, x : \sigma, y : \tau, \overline{w} : \overline{\rho}_2 \vdash C]\!] \in R^{\theta}_{\overline{\rho}_1 \sigma \tau \overline{\rho}_2}$. Then $[\![\overline{z} : \overline{\rho}_1, y : \tau, x : \sigma, \overline{w} : \overline{\rho}_2 \vdash C]\!] \in R^{\theta}_{\overline{\rho}_1 \sigma \tau \overline{\rho}_2}$ is also in R, and this is just the result of applying switch to the former function. So R is closed under switch. **merge** and **vac** are established similarly.

For **abs** we must show that the structure is λ-closed. Suppose that $[\![\Gamma, x : \sigma \vdash C : \tau]\!]$ is a Russellian function. We will define $\lambda[\![\overline{y} : \overline{\rho}, x : \sigma \vdash C : \tau]\!]$ to be $[\![\Gamma \vdash \lambda x.C : \sigma \to \tau]\!]$. First observe that $\mathbf{app}([\![\Gamma \vdash \lambda x.C : \sigma \to \tau]\!][\overline{A}], [B]) = \mathbf{app}([(\lambda x.C[\overline{A}/\overline{y}]), [B]) = [(\lambda x.C[\overline{A}/\overline{y}])B] = [C[\overline{A}B/\overline{y}x]] = [\![\overline{y} : \overline{\rho}, x : \sigma \vdash C : \tau]\!][\overline{A}][B]$. This establishes that condition $\mathbf{app} \circ (\lambda \mathbf{r} \times \mathbf{id}) = \mathbf{r}$. Next we need to show that this is the unique such logical form. Suppose that $\mathbf{app}([\![\overline{y} : \overline{\rho} \vdash M : \sigma \to \tau]\!][\overline{A}], [B]) = [\![\overline{y} : \overline{\rho}, x : \sigma \vdash C : \tau]\!][\overline{A}][B] = [(\lambda x.C[\overline{A}/\overline{y}])B]$ for all $\overline{A}B$. Then because T is witness complete we have $\forall_{\sigma} \overline{y} x.(Mx =_{\tau} (\lambda x.C)x) \in T$. By Formal Functionality and η it follows that $M =_{\sigma \to \tau} \lambda x.C \in T$ so that $[M] = [\lambda x.C]$ as required.

Next we show that $[\![\cdot]\!]$ satisfies the environment model condition. First we will establish Compositionality$_1$. Suppose that $[\![\overline{x} : \overline{\rho} \vdash A]\!] = [\![\overline{x} : \overline{\rho} \vdash B]\!]$. This means $[A[\overline{D}/\overline{x}]] = [\overline{B}[\overline{D}/\overline{x}]]$ for all type apt sequences of terms $\overline{D}$, and

by the witness completeness of T'' this means that $\forall\overline{x}(A =_\tau B)\,T''$. By Formal Functionality, $C = C \to C = C[B/A] \in T''$, and by MP and Refl $C = C[B/A] \in T''$. So $[C] = [C[B/A]]$ as required.

For Compositionality$_2$, we apply $[\![\overline{yzy}' : \overline{\rho\theta\rho'} \vdash M[N/x]]\!]$ and $[\![\overline{y}x\overline{y}' : \overline{\rho}\sigma\overline{\rho}' \vdash M]\!] \circ (\overline{\mathbf{id}} \times [\![\overline{z} : \overline{\theta} \vdash N]\!] \times \overline{\mathbf{id}})$ to an appropriate sequence $[\overline{ABA}']$. In the former case we obtain $[M[N/x][\overline{ABA'}/\overline{yzy'}]]$ and in the latter we get $[M[\overline{AA'}/\overline{yy'}][N[\overline{B}/\overline{z}]/x]]$. Because $\overline{y}x\overline{y}'$ are all the free variables in M, and $\overline{z}$ all the variables in N, and these don't overlap, these substitutions commute and we end up with the same term.

The other conditions App, Conc, Rules, Abs, Id, Vac, Merge, and Switch are pretty much straightforward. For instance, for Concretion we can reason from right to left: $\mathbf{app}([\![\overline{y} : \overline{\rho} \vdash M : \sigma \to \tau]\!][\overline{A}], [B]) = (M[\overline{A}/\overline{y}])B = Mx[\overline{A}B/\overline{y}x] = [\![\overline{y} : \overline{\rho}, x : \sigma \vdash Mx : \tau]\!][\overline{A}][B]$. So $\mathbf{app} \circ ([\![\Gamma \vdash M : \sigma \to \tau]\!] \times \mathbf{id}) = [\![\Gamma, x : \sigma \vdash M : \tau]\!]$ as required.

Lastly we must show that it is a model. The clauses for the logical operations $\to$, $\forall$ and $=$ are standard and follow from the negation completeness and the witness completeness of T''. The key thing to check is the clause for definability sentences. For each closed definability sentence, $\vdash A \rhd_{\overline{x}} B$, we must show that: $v([\![\overline{A} \rhd_{\overline{x}} B]\!]^g) = 1$ (i.e. $\overline{A} \rhd_{\overline{x}} B \in T''$) if and only if there exists a sequent $\overline{x} : \overline{\rho} \vdash C : \tau$ of $\mathcal{L}^-(\Lambda, \Xi, \emptyset)$ such that $[\![\overline{x} : \overline{\rho} \vdash C : \tau]\!]([\![A_1]\!]^{g'}, \ldots, [\![A_n]\!]^{g'}) = [\![B]\!]^{g'}$ for every variable assignment $g'[\overline{x}]g$.

Now if $\overline{A} \rhd_{\overline{x}} B \in T''$ it follows that for some witnessing rule r_n, $\forall\overline{x}(r_n(\overline{A}) = B) \in T''$, and thus $v([\![\vdash \forall\overline{x}(r_n(\overline{A}) = B)]\!]^g) = 1$. Since $r_n(\overline{A}) = B$ contains only $\overline{x}$ free, this means that for all variable assignments $g'[\overline{x}]g$, $v([\![r_n(\overline{A}) = B]\!]^{g'}) = 1$ so that $[\![r_n(A_1 \ldots A_n)^{g'}]\!] = [\![B]\!]^{g'}$ for all relevant sequences. Thus $[\![\overline{x} : \overline{\sigma} \vdash r_n(\overline{x})]\!]([\![A_1]\!]^{g'}, \ldots, [\![A_n]\!]^{g'}) = [\![B]\!]^{g'}$ for all such g', so $[\![\overline{x} : \overline{\sigma} \vdash r_n(\overline{x})]\!]$ is the required logical form.

Conversely, suppose there exists a sequent $y_1 : \sigma_1 \ldots y_n : \sigma_n \vdash C : \tau$ such that $[\![\overline{x} : \overline{\rho} \vdash C : \tau]\!]([\![A_1]\!]^{g'}, \ldots, [\![A_n]\!]^{g'}) = [\![B]\!]^{g'}$ for every variable assignment $g'[\overline{x}]g$. By Compositionality the left hand side of the required identity is the same as $[\![C[\overline{A}/\overline{x}]]\!]^{g'}$. By the semantic clause for equality, this means that $v([\![\forall\overline{x}(C[\overline{A}/\overline{y}] = B]\!]^{g'})) = 1$, and thus that $\forall\overline{x}(C[\overline{A}/\overline{y}] = B) \in T$. By lemma 5.1, since $y_1 : \sigma_1 \ldots y_n : \sigma_n \vdash C$ is in $\mathcal{L}^-(\Lambda, \Xi, \emptyset)$, $\overline{y} \rhd_{\overline{y}} C \in T$, and then by Instantiation $\overline{A} \rhd_{\overline{x}_1 \ldots \overline{x}_n} C[A_1/y_1 \ldots A_n/y_n] \in T$. Since the above universally quantified identity is in T, we infer the desired result that $\overline{A} \rhd_{\overline{x}_1 \ldots \overline{x}_n} B \in T$ using Formal Functionality.

□

5.4 The logic of conditional definability

Here I have a less complete picture. But I will present some partial results.

Let's begin with some principles valid in all conditional definability structures. Since $(D, R[X], \mathbf{app})$ is a definability structure for any choice of X all Base Principles hold of $\rhd^{\Sigma}$ when Σ is held fixed. Additionally we have the following principles are valid

Reflexivity $\rhd^{\{A\}} A$

Weakening $\Gamma \rhd^{\Sigma}_{\overline{x}} A \to \Gamma \rhd^{\Sigma'}_{\overline{x}} A$ whenever $\{\Sigma\} \subseteq \{\Sigma'\}$

Upper Cut $(\Gamma \rhd^{\Sigma}_{\overline{x}} A) \to (\Delta, A, \Delta' \rhd^{\Sigma'}_{\overline{y}} B) \to (\Delta, \Gamma, \Delta' \rhd^{\Sigma\Sigma'}_{\overline{xy}} B)$

Lower Cut $(\Gamma \rhd^{\Sigma}_{\overline{x}} A) \to (\Delta \rhd^{\Sigma' A}_{\overline{y}} B) \to (\Delta \rhd^{\Gamma\Sigma\Sigma'}_{\overline{xy}} B)$.

Proposition 5.4. *The following may be derived from the principle listed above.*

Lowering $\Gamma, A, \Delta \rhd_{\Sigma} B \to \Gamma, \Delta \rhd_{\Sigma A} B$

Monotonicity $\Gamma \rhd^{\Sigma} A \to \Delta \rhd^{\Sigma\Gamma A} B \to \Delta \rhd^{\Sigma\Gamma} B$

Lowering follows from Upper Cut using an instance of Refl ($\rhd_{\{A\}} A$) as a premise, letting $\Sigma_1 = \{A\}$ and letting the remaining Greek letters denote the empty sequence. Monotonicity is also an instance of Lower Cut.

Although I have not worked through the details, I believe the principles listed are complete for the logic of all conditional definability structures.

What about the logic of *minimal* conditional definability structures. Here I have some observations that produce axiomatizations in the cases where $\mathbf{id}, \mathbf{merge} \in \Lambda$. Consider first the full λ-signature $\{\mathbf{id}, \mathbf{vac}, \mathbf{switch}, \mathbf{merge}, \mathbf{abs}\}$. We can completely axiomatize the minimal conditional definability structure, because in that structure conditional definability is definable from unconditional definability, via the following reduction:

Reduction(switch, vac) $\Gamma \rhd^{\Delta}_{\overline{x}} A \leftrightarrow \Gamma, \Delta \rhd_{\overline{x}} A$

Notice that in the presence of Lowering the right-to-left direction is already derivable, and is consequently true in any conditional definability structure. The left-to-right direction is specific to minimal conditional definability structures. By adding this to our existing axiomatization of unconditional logical definability we obtain a sound and complete axiomatization of the logic of minimal conditional $\{\mathbf{id}, \mathbf{vac}, \mathbf{switch}, \mathbf{merge}, \mathbf{abs}\}$-structures.

Here is the informal reason this reduction works. Suppose that there is a propositional function constructed from constituents in Δ that can be filled with the constituents Γ to get A. We can then poke the constituents from Δ out of that propositional function in a certain way to get a logical form which can then be filled with Γ and Δ together to get A. Because we are working in the full λ-signature, the order and the number of the constituents from Δ in the original propositional function doesn't matter, because there is a suitable logical form which will reorder and duplicate elements of Δ, in the sequence it is presented, as needed.

More generally, in the cases where $\mathbf{merge} \in \Lambda$ (and still assuming $\mathbf{id} \in \Lambda$) $\mathbf{L}[\Lambda]$ can completely be axiomatized by adding to our axiomatization of $\mathbf{L}[\Lambda]$ the axiom:

Reduction $\Gamma \rhd^{\Delta}_{\overline{x}} A \leftrightarrow \bigvee_{\{\Sigma\} \subseteq \{\Gamma\Delta\}} \Sigma \rhd_{\overline{x}} A$ where Σ ranges over sequences with no-repetitions.

Σ meets the requirement above when Σ is of the form $\Delta_1 A_1 \ldots \Delta_n A_n \Delta_{n+1}$ where each of $\Delta_1 \ldots \Delta_n$ only contain elements from Δ (i.e. $\{\Delta_i\} \subseteq \{\Delta\}$, $\overline{A} : \overline{\sigma}$ is a subsequence of Γ and the sequence contains no repetitions. Because there are only finitely many sequences like this, this disjunction is finite. The reason this axiom suffices in the presence of **merge** is because we can see that in any minimal conditional definability structure satisfying **merge**, $\Gamma \rhd^{\Delta} A$ holds if and only if $\Sigma \rhd A$ holds for some Σ of the required form: see proposition 5.5 below. However, in definability structures in which **merge** does not hold, the restriction to sequences without repetitions is not sufficient, and so there is no finite disjunction that can capture the force of the conditional definability structure.

As we saw above, this axiom can be simplified in the presence of the further axioms Switch and Vac. In the case that we have the axioms Switch and Vac, this disjunction is equivalent to $\Gamma, \Delta \rhd A$, and so we can instead use the simple axiom Reduction($\mathbf{switch}, \mathbf{vac}$) stated above. In the case that we have only Switch or only Vac, this big disjunction is equivalent to a disjunction with a smaller number of disjuncts. Thus this axiom suffices to

capture the reduction for the logic of any λ-signature, although the force of the axiom is different depending on the background logic of unconditional definability and will often be replaceable with a more transparent axiom.

The following proposition establishes the soundness of Reduction in all minimal conditional Λ-definability models.

Proposition 5.5.

1. *Suppose that Λ is the full gap signature $\{\mathbf{vac}, \mathbf{merge}, \mathbf{switch}, \mathbf{id}, \mathbf{abs}\}$, that $(D, R, \mathbf{app})$ is a Λ-definability structure (and is conditionally λ-closed if $\mathbf{abs} \in \Lambda$), and that $\overline{d} \in D^{\overline{\theta}}$. Then if $\mathbf{a} \in (R^{\Lambda}[\{\overline{d}\}])^{\sigma}_{\overline{\rho}}$, there is a $\mathbf{r_a} \in R^{\sigma}_{\overline{\theta}\overline{\rho}}$ such that $\mathbf{r_a}(\overline{de}) = \mathbf{a}(\overline{e})$ for every $\overline{e} \in D^{\overline{\rho}}$.*

2. *More generally, suppose that, that $(D, R, \mathbf{app})$ is a Λ-definability structure, and that $\overline{d} \in D^{\overline{\theta}}$. Then if $\mathbf{a} \in (R^{\Lambda}[\{\overline{d}\}])^{\sigma}_{\overline{\rho}}$, there is a $\mathbf{r_a} \in R^{\sigma}_{\overline{\theta}\overline{\rho}}$ and collection of sequences $\overline{d}_1 ... \overline{d}_{n+1}$ whose elements are all drawn from $\overline{d}$ such that $\mathbf{r_a}(\overline{d_1 e_1 ... d_n e_n d_{n+1}}) = \mathbf{a}(\overline{e_1 ... e_n})$ for every $\overline{e_1 ... e_n} \in D^{\overline{\rho}}$.*

Proof. We prove 1. Say that a function $\mathbf{a} : D^{\overline{\rho}} \to D^{\sigma}$ is *reducible* when there exists a $\mathbf{r} \in R^{\sigma}_{\overline{\rho}'\overline{\rho}}$ such that $\mathbf{r}(\overline{d}\overline{e}) = \mathbf{a}(\overline{e})$ for all $\overline{e} \in D^{\overline{\rho}}$. Let S be the set of reducible functions. Recall that $R^{\Lambda}[\overline{d}]$ is the smallest set of functions containing $\{\overline{d}^*\}$ closed under the operations of concretion ($\mathbf{r} \mapsto \mathbf{app} \circ (\mathbf{r} \times \mathbf{id})$), parallel composition ($(\mathbf{r}, \mathbf{p}) \mapsto \mathbf{r} \circ (\overline{\mathbf{id}} \times \mathbf{p} \times \overline{\mathbf{id}})$), $\mathbf{id}, \mathbf{vac}, \mathbf{merge}, \mathbf{switch}$ and λ. Thus, in order to show that $R[\overline{d}] \subseteq S$, it suffices to show that S contains $\{\overline{d}^*\}$ and is closed under the operations of concretion, parallel composition, $\mathbf{id}, \mathbf{vac}, \mathbf{merge}, \mathbf{switch}$ and λ

(i) We must show that $\mathbf{a} = * \mapsto d_i$ with d_i in $\overline{d}$ is reducible. Since $\mathbf{id}, \mathrm{vac} \in X$, $\mathbf{id} \in R^{\theta_i}_{\theta_i}$ and so $\mathbf{id} \circ \mathrm{vac}_{\overline{\theta}\theta_i} \in R^{\theta_i}_{\overline{\theta}}$ and $\mathrm{vac}_{\overline{\theta}\theta_i}(\overline{d}) = d_i$ as required.

(ii) Suppose that $\mathbf{a} : D^{\overline{\rho}_1 \sigma \overline{\rho}_3} \to D^{\tau}$ and $\mathbf{b} : D^{\overline{\rho}} \to D^{\sigma}$ are reducible. So there exists $\mathbf{r_a}, \mathbf{r_b}$ in R, such that $\mathbf{r_a}(\overline{de_1 ce_3}) = \mathbf{a}(\overline{e_1 ce_3})$, and $\mathbf{r_b}(\overline{de_2}) = \mathbf{b}(\overline{e}_2)$.

Then $\mathbf{a}(\overline{e}_1 \mathbf{b}(\overline{e}_2)\overline{e}_3) = \mathbf{r_a}(\overline{de_1}\mathbf{r_b}(\overline{de_2})\overline{e}_3)$. We can construct $\mathbf{r}_{\mathbf{a} \circ (\overline{\mathbf{id}} \times \mathbf{b} \times \overline{\mathbf{id}})}$ as $\mathbf{r_a} \circ (\overline{\mathbf{id}} \times \mathbf{r_b} \times \overline{\mathbf{id}}) \circ \overline{\mathrm{merge}}$. Note that we need to postcompose with a suitable sequence of merges: if we don't it won't map the sequence $\overline{de_1 de_2 e_3}$ to the desired output $\overline{de_1 e_2 e_3}$.

(iii) Suppose $\mathbf{a} : D^{\overline{\rho}} \to D^{\sigma \to \tau}$ is reducible. So there is some $\mathbf{r_a}$ in R such that $\mathbf{r_a}(\overline{de}) = \mathbf{a}(\overline{e})$. We want $\mathbf{r}_{\mathbf{app} \circ (\mathbf{a} \times \mathbf{id})}(\overline{dec}) = \mathbf{app}(\mathbf{a}(\overline{e}), c)$, which can be obtained by letting $\mathbf{r}_{\mathbf{app} \circ (\mathbf{a} \times \mathbf{id})}$ be $\mathbf{app} \circ (\mathbf{r_a} \times \mathbf{id})$. Since this is in R, $\mathbf{app} \circ (\mathbf{a} \times \mathbf{id})$ is reducible.

(iv) Suppose that $\mathbf{a} \in D^{\overline{\rho}\sigma} \to D^{\tau}$ is reducible, and $\mathbf{r_a}(\overline{de}c) = \mathbf{a}(\overline{e}c)$ for every $\overline{de}c$, where $\mathbf{r_a}$ is in R. Then by λ-closure $\lambda\mathbf{r_a}$ is also in R and has the property that $\mathbf{app}(\lambda\mathbf{r_a}(\overline{de}), c) = \mathbf{r_a}(\overline{de}c) = \mathbf{a}(\overline{e}) = \mathbf{app}(\lambda\mathbf{a}(\overline{e}), c)$ for every $\overline{de}c$. This means $\mathbf{app} \circ (\lambda\mathbf{r_a} \times \mathbf{id})(\overline{d}) = \mathbf{a} = \mathbf{app} \circ (\lambda\mathbf{a} \times \mathbf{id})$. Note that since $\lambda\mathbf{r_a} \in R$, $\lambda\mathbf{r_a}(\overline{d}) \in R[\{\overline{d}\}]$ since $\lambda\mathbf{r_a}(\overline{d})$ can be obtained by composing $\mathbf{r_a}$ with the functions $* \mapsto d_i$. Now, since $\lambda\mathbf{a}$ is the unique Russellian function in $R[\{\overline{d}\}]$ such that $\mathbf{app} \circ (\lambda\mathbf{a})(\overline{e}) \times \mathbf{id}) = \mathbf{a}$ (because we have assumed that R is conditionally λ-closed), it follows that $\lambda\mathbf{r_a})(\overline{d}) = \mathbf{a}$.

Checking the cases for vac, switch and merge is straightforward. In each case we assume that $\mathbf{a}$ is reducible, is represented by $\mathbf{r_a}$ in R such that $\mathbf{r_a}(\overline{de}) = \mathbf{a}(\overline{e})$, and we set $\mathbf{r_{a \circ \mathrm{vac}}}) = \mathbf{r_a} \circ \mathrm{vac}$, $\mathbf{r_{a \circ \mathrm{merge}}}) = \mathbf{r_a} \circ \mathrm{merge}$, and $\mathbf{r_{a \circ \mathrm{switch}}}) = \mathbf{r_a} \circ \mathrm{switch}$, choosing appropriate type-subscripts, to obtain their reductions and establish they are reducible too.

The proof of 2 is similar, except one must be more careful about the number and positioning of $\overline{d}$ in the arguments of $\mathbf{r_a}$. □

5.5 Some Remarks on Logics without Identity

We have set aside the case of logics without Identity, i.e. $\mathbf{L}[\Lambda]$ where $\mathbf{id} \notin \Lambda$ because it requires several modifications to the formalism. There are two sources of complications that make the results of the last section hard to generalize. One is that variables on their own are not terms of $\mathcal{L}(\Lambda, \Xi, \Sigma)$ when $\mathbf{id} \notin \Lambda$, requiring more complications in the syntax of $\rhd$, and more care over the statement of the axioms and rules. The second complication is that primitive syncategorematic rules are not grammatically intersubstitutible with arbitrary propositional functions, meaning we need to have a stronger version of Rule Gen. Here I briefly described the complications, and outline a strategy for generalizing the completeness results of the previous section.

Observe that in order to form the sequent $\vdash x \rhd_x^{\sigma} x$ we would need to begin with two instances of the Identity sequent $x : \sigma \vdash x : \sigma$. In general, in order to be able to form terms of the form $A_1, \ldots, A_n \rhd_{\overline{x}} B$ where some of $A_1, \ldots, A_n, B$ are variables we need to use the more general rule for forming $\rhd$ terms described at the end of section 2.3.

$$\frac{\Gamma_1 \vdash A_1 \qquad \ldots \qquad \Gamma_n \vdash A_n \qquad \Sigma \vdash B}{\Delta \vdash t_1 \ldots t_k \rhd_{\overline{x}} B}$$

where Δ is the sequence $s_1 \ldots s_k$ where s_i is either a variable or a sequence

Γ_j, and t_i is s_i if s_i is a variable, and t_i is A_i if s_i is Γ_i. Given this rule, $x \rhd^{\sigma}_{x} x$ is a well-formed sentence, even though x is not a term.

The second complication is that, in the absence of identity, primitive syncategorematic rules, r, do not always determine (lingistic) propositional functions, because the sequent of the form $\overline{x} : \overline{\sigma} \vdash r\overline{x}$ may not be part of the language. So Rule Gen seems too weak: we may be unable to establish of an arbitrary primitive rule, r, that $\neg\forall\overline{x}(r(\overline{A}) = B)$, when this might hold for an arbitrary propositional function.

This is because it is possible to form a linguistic propositional function of the form $\overline{\Gamma} \vdash r(\overline{A})$ if we have already constructed the propositional functions $\Gamma_i \vdash A_i$ for $i = 1 \ldots n$, but it is not possible to form a propositional function of the form $\overline{x} : \overline{\sigma} \vdash r(\overline{x})$ using bare variables without Identity. This means that the rule Rule Gen needs to be formulated carefully. Thus we should introduce propositional function variables as distinct from primitive rules, that can have an arbirary logical form. This means we add a new sequent rule for propositional function variables.

$$\frac{}{x_1 : \sigma_1 \ldots x_n : \sigma_n \vdash p(x_1, \ldots, x_n)}$$

Without Identity, Rules does not imply the new rule, nor conversely. With identity Rules implies the above, since it follows from the premises $x_i \vdash x_i$ $i = 1 \ldots n$. The reverse implication is not possible without taking the substitution lemma as instead a primitive rule; in the present case it is an admissible rule.

The proposed axiomatization of $\mathbf{L}[\Lambda]$ when $\mathbf{id} \notin \Lambda$ is described as before with the following exceptions. Firstly, the axiom Id, $x \rhd_x x$ must be removed. Second, Rule Gen needs to be formulated using variables for arbitrary propositional functions, rather than primitive rules. Note that we are not to eliminate the propositional function variables when $\mathbf{abs} \in \Lambda$ in the way described in section 5.2 because Decomposition is not a well-formed sentence without Id since it involves quantification into predicating position. The propositional function variables will play the role of the witnessing rules in the proof of the completeness theorem since it is quantification over propositional functions, not the primitive rules, that is used in the truth clauses for metaphysical definability.

A The Substitution Lemma

Lemma A.1 (The substitution lemma)**.** *If* $\Gamma, x : \sigma, \Gamma' \vdash M : \tau$ *and* $\Delta \vdash N : \sigma$ *are derivable in* $\mathcal{L}(\Lambda, \Xi, \Sigma)$*, and* Δ *has no variables in common with* $\Gamma, x : \sigma, \Gamma'$*, then* $\Gamma, \Delta, \Gamma' \vdash M[N/x] : \tau$ *is also derivable in* $\mathcal{L}(\Lambda, \Xi, \Sigma)$*.*

Proof. Pick a fixed proof, D, of $\Gamma, x : \sigma, \Gamma' \vdash M : \tau$. We can transform it into a proof of $\Gamma, \Delta, \Gamma' \vdash M[N/x] : \tau$ in the following way.

The guiding idea in this argument is that it is possible to trace the variable $x : \sigma$ backwards through the proof of $\Gamma, x : \sigma, \Gamma' \vdash M : \tau$, being careful about the fact that the rule merge relabels variables, and systematically replace variables tracing back from x with a copy of Δ, and by replacing free occurrences of these variables on the right-hand-side of the sequent suitable copies of N.

For each variable y let Δ_y be a variant of Δ constructed from new variables of matching type: we require that Δ_y is disjoint from Δ_z for $y \neq z$ and not containing any variables appearing in the derivation of $\Gamma, x : \sigma, \Gamma' \vdash M : \tau$ and $\Delta \vdash N : \sigma$. Let E_y be the proof of $\Delta_y \vdash N_y : \tau$ obntained by relabeling the variables in the proof of $\Delta \vdash N : \tau$, and N_y is the resultant term introduced by this derivation.

Let $Y = \{y_1, \ldots, y_n\}$ be a set of variables. By Γ^Y we shall mean the sequence obtained by replacing each occurrence of y in Γ with Δ_y for each variable$y \in Y$, and by M^Y we mean the result of replacing all free occurrences of y with N_y in M, for $y \in Y$.

We can now describe a transformation of a derivation D to D^Y as follows.

app $(\frac{D_1 \quad D_2}{\Sigma \vdash M:\tau})^Y = \frac{D_1^Y \quad D_2^Y}{\Sigma^Y \vdash M^Y}$

abs, conc

sync $(\frac{r \in \Sigma, D_1 \quad \ldots \quad D_n}{\Gamma \vdash r(M_1,\ldots,M_n):\sigma})^Y = \frac{r \in \Sigma, D_1^Y \quad \ldots \quad D_n^Y}{\Gamma \vdash r(M_1,\ldots,M_n)^Y:\sigma}$

id $(\frac{}{x:\sigma \vdash x:\sigma})^Y = \frac{E_y}{\Delta^Y \vdash N(\frac{D}{\Sigma \vdash M:\tau})^Y = \frac{D^Y}{\Sigma^Y \vdash M^Y}_y}$ if $y \in Y$, and is $\frac{}{x:\sigma \vdash x:\sigma}$ otherwise.

This leaves only the case for merge to describe. when $z \notin Y$ we the translation simply commutes as in the rules vac, switch, abs and conc. In the case that

$z \in Y$ it is given as follows.

$$(\frac{\dfrac{D}{\Gamma, x:\sigma, \Sigma, y:\sigma, \Lambda \vdash M:\tau}}{\Gamma, z:\sigma, \Sigma, \Lambda \vdash M[z/x][z/y]:\tau})^Y = \frac{\dfrac{D^{Y\cup\{x,y\}}}{(\Gamma, x:\sigma, \Sigma, y:\sigma, \Lambda)^{Y\cup\{x,y\}} \vdash M^{Y\cup\{x,y\}}:\tau}}{\vdots}}{(\Gamma, z:\sigma, \Sigma, \Lambda)^Y \vdash (M[z/x][z/y])^Y : \tau}$$

Since $(\Gamma, x:\sigma, \Sigma, y:\sigma, \Lambda)^{Y\cup\{x,y\}} = (\Gamma^Y, \Delta_x, \Sigma^Y, \Delta_y, \Lambda^Y)$. So here the verticals elipses are filled with a series of applications of merge that successively merge the first, second, third etc. variables of Δ_x and Δ_y into the first, second, third, etc variables of Δ_z.

The translations of vac and switch are subject to a similar issue, and must be defined as follows

$$(\frac{\dfrac{D}{\Gamma, x:\sigma, y:\tau, \Gamma' \vdash M:\tau}}{\Gamma, y:\tau, x:\sigma, \Gamma' \vdash M:\tau})^Y = \frac{\dfrac{D^Y}{\Gamma^Y, \Delta_x, \Delta_y, \Gamma'^Y \vdash M^Y:\tau}}{\vdots}}{\Gamma^Y, \Delta_y, \Delta_x, \Gamma'^Y \vdash M^Y : \tau}$$

here the $\vdots$ represent a series of applications of switch that can transform the sequence Δ_x, Δ_y into Δ_y, Δ_x.

$$(\frac{D}{\Sigma \vdash M:\tau})^Y = \frac{D^Y}{\Sigma^Y \vdash M^Y}$$

we might get a transition like $\frac{\Gamma,\Delta_x,\Delta_y,\Gamma'\vdash M}{\Gamma,\Delta_y,\Delta_x,\Gamma'\vdash M}$ which must be obtained by several applications of switch, rather than one. In the case of vac you might get a transition like $\frac{\Gamma,\Sigma\vdash M:\sigma}{\Gamma,\Delta,\Sigma\vdash M:\sigma}$ which is obtained by repeatedly applying vac, one application for each variable in Δ.

It can be seen by inspection that the transformation maps a derivation to a derivation. Finally, it follows that be seen that if D is a derivation $\Gamma, x:\sigma, \Gamma' \vdash M:\tau$, then $D^{\{x\}}$ is a derivation, by the above, and that it is a derivation of $\Gamma, \Delta, \Gamma' \vdash M[N/x]:\tau$ by the way the translation is defined.

The result is not quite a proof. For instance, when we apply the transformation to an instance of merge we get the transition $\Gamma, \Delta, \Gamma', \Delta, \Gamma'' \vdash M:\tau$ to $\Gamma, \Delta, \Gamma', \Gamma''v \vdash M:\tau$ which is not an instance of merge, but can be obtained by a sequence of applications of merge. Similarly, instances of switch and vac transform to transitions that can be obtained by a sequence of switches, and vacs.

□

References

Andrew Bacon. The broadest necessity. *Journal of Philosophical Logic*, 47 (5):733–783, 2018. doi: 10.1007/s10992-017-9447-9.

Andrew Bacon. Substitution structures. *Journal of Philosophical Logic*, 48 (6):1017–1075, 2019. doi: 10.1007/s10992-019-09505-z.

Andrew Bacon. Logical combinatorialism. *Philosophical Review*, 129(4):537–589, 2020. doi: 10.1215/00318108-8540944.

Andrew Bacon. A theory of structured propositions. *Philosophical Review*, 132(2):173–238, 2023a. doi: 10.1215/00318108-10294409.

Andrew Bacon. *A Philosophical Introduction to Higher-Order Logics*. Routledge, 2023b.

Andrew Bacon and Cian Dorr. Classicism. In Peter Fritz and Nicholas K. Jones, editors, *Higher-order Metaphysics*. Oxford University Press, 2024.

Francis Herbert Bradley. *Appearance and Reality: A Metaphysical Essay*. London, England, 1893.

Cian Dorr. Non-symmetric relations. In Dean Zimmerman, editor, *Oxford Studies in Metaphysics Volume 1*, pages 155–92. Oxford University Press, 2004.

Cian Dorr. To be f is to be g. *Philosophical Perspectives*, 30(1):39–134, 2016. doi: 10.1111/phpe.12079.

Kit Fine. Neutral relations. *Philosophical Review*, 109(1):1–33, 2000. doi: 10.1215/00318108-109-1-1.

Kit Fine. The pure logic of ground. *Review of Symbolic Logic*, 5(1):1–25, 2012. doi: 10.1017/s1755020311000086.

Peter Fritz. Ground and grain. *Philosophy and Phenomenological Research*, 105(2):299–330, 2021. doi: 10.1111/phpr.12822.

Jeremy Goodman. Reality is not structured. *Analysis*, 77(1):43–53, 2017. doi: 10.1093/analys/anw002.

Jeremy Goodman. Agglomerative algebras. *Journal of Philosophical Logic*, 48(4):631–648, 2018. doi: 10.1007/s10992-018-9488-8.

Joachim Lambek. Deductive systems and categories ii. standard constructions and closed categories. In *Category Theory, Homology Theory and their Applications I: Proceedings of the Conference held at the Seattle Research Center of the Battelle Memorial Institute, June 24–July 19, 1968 Volume One*, pages 76–122. Springer, 1968.

John C Mitchell. *Foundations for programming languages*, volume 1. MIT press Cambridge, 1996.

John Myhill. Problems arising in the formalization of intensional logic. *Logique Et Analyse*, 1(1):78–83, 1958. doi: 10.2307/2272577.

F. P. Ramsey. On facts and propositions. In Simon Blackburn and Keith Simmons, editors, *Truth*. Oxford University Press, 1999.

Greg Restall. *An Introduction to Substructural Logics*. Routledge, 2000.

Bertrand Russell. *Principles of Mathematics*. Routledge, New York,, 1903.

Bertrand Russell. *The Philosophy of Logical Atomism*. Open Court, 1940.

Gabriel Uzquiano. A neglected resolution of russell?s paradox of propositions. *Review of Symbolic Logic*, 8(2):328–344, 2015. doi: 10.1017/s1755020315000106.

Timothy Williamson. Converse relations. *Philosophical Review*, 94(2):249–262, 1985. doi: 10.2307/2185430.

Ludwig Wittgenstein. *Tractatus Logico-Philosophicus (Trans. Pears and Mcguinness)*. Routledge, New York,, 1921.