J Philos Logic (2017) 46:65–95
DOI 10.1007/s10992-016-9394-x

Same Same But Different: An Alphabetically Innocent Compositional Predicate Logic

Udo Klein[1] · **Wolfgang Sternefeld**[2]

Received: 10 June 2014 / Accepted: 24 January 2016 / Published online: 5 March 2016

1 Introduction

1.1 Motivation

Compositionality is at the heart of model theoretical semantics and its application to the semantics of natural language. As has become standard practice, linguists translate a fragment of English into an intensional extension of classical predicate logic (PL). Yet, somewhat ironically and strangely, PL itself is not compositional, because the standard truth conditions for quantificational statements are not a function of the denotations of its parts but depend on value assignments for variables. This kind of dependence on value assignments leads to non-compositionality as will be demonstrated explicitly in Section 1.2. One could, as is well-known, remedy this awkwardness by considering not truth values as denotations of formulas but sets of value assignments for variables. As we will show in Section 1.3, such a semantics is compositional but not "alphabetically invariant" (or "innocent"). In this article we will formulate a compositional extension of PL that is both compositional and innocent.

"Same Same But Different" is a hackneyed saying used in Thailand to convey that differences do not always matter; it has also become known as the title of a movie by Detlev Buck. We would like to thank Marcus Kracht, Ede Zimmermann and anonymous reviewers for comments and valuable suggestions.

✉ Udo Klein
udomichaelklein@googlemail.com

Wolfgang Sternefeld
sternefeld@uni-tuebingen.de

1 PITSS GmbH, Otto Brenner Str. 209, 33604 Bielefeld, Germany

2 Uni Tübingen, Seminar für Allgemeine Sprachwissenschaft, Tübingen, Germany

The lack of alphabetical "innocence" (or "invariance") results from the fact that the denotation of $P(x)$ is different from that of $P(y)$ although the formulas are mere alphabetic variants of each other: Let $[\![.]\!]$ be the interpretation function for formulas, and I the interpretation function for predicates, and g a variable for assignment functions. Then $[\![P(x)]\!] = \{g : g(x) \in I(P)\}$ and $[\![P(y)]\!] = \{g : g(y) \in I(P)\}$. Let g be an assignment such that $g(x) \in I(P)$ and $g(y) \notin I(P)$. Then $[\![P(x)]\!] \neq [\![P(y)]\!]$.

This problem is related to what [2], p. 7, calls the "antinomy of the variable":

> Suppose that we have two variables, say "x" and "y"; and suppose that they range over the same domain of individuals, say the domain of all real numbers. Then it appears as if we wish to say contradictory things about their semantic role. For when we consider their semantic role in two distinct expressions – such as "$x > 0$" and "$y > 0$," we wish to say that it is the same. Indeed, this would appear to be as clear a case as one could hope to have of a merely "conventional" or "notational" difference; the difference is merely in the choice of the symbol and not at all in linguistic function. On the other hand, when we consider the semantic role of the variables in the same expression – such as in "$x > y$" – then it seems equally clear that it is different. Indeed, it would appear to be essential to the semantic role of the expression as a whole that it contains two distinct variables, not two occurrences of the same variable, and presumably this is because the roles of the distinct variables are not the same.[1]

Another conceptual problem results from making assignment functions part of a compositional semantics (cf. Section 1.3 for a formal system). Once denotations are compositionally defined in terms of assignment functions (cf. [4]), these functions become part of the ontology, with the undesirable consequence that there is more in our ontology than the simple denotations found in the standard semantics. In particular, the semantics of a language has to refer to the variables of the language and thereby becomes language dependent. Calling ordinary denotations "local extensions" and sets of assignment functions "global extensions", [14], p. 243 argue that the regained compositionality is particularly problematic for natural language semantics:

> Apart from the fact that global extensions—or any of their substitutes—are rather unwieldy, they are also somewhat of a cheat. For other than ordinary extensions, which correspond to the objects referred to in the non-linguistic world, global extensions are language dependent in that they are functions whose domain is the set of variable assignments which in turn are functions defined on variables, [. . .] and hence linguistic expressions.

But do compositionality and alphabetical innocence necessarily exclude each other? Can we provide for an interpretation of predicate logic (with open formulas) that is both compositional and alphabetically innocent but does not involve unwarranted ontological commitments? In this paper we will propose and discuss variants

[1]For a more detailed exposition of the problem, see Chapter 1 of [2].

of PL (with open formulas) which are indeed compositional and alphabetically innocent, ie. do not invoke reference to variables in their ontology.

Before going into the details of our proposal, let us briefly review some elementary notions like compositionality and alphabetic innocence, demonstrating thereby why the classical systems have the properties that crucially motivate an alternative approach.

1.2 Assignment Functions and the Lack of Compositionality

To state our point explicitly, we briefly state the syntax and semantics of PL as standardly found in textbooks. We omit constants as they are not relevant for the argument. Our language thus consists of (a) a finite set of relational symbols $\{R_i : 1 \leq i \leq n\}$, each of a finite adicity; (b) an infinite set of variables $\{x, y, x', x'', x''', \ldots\}$; (c) the connectors: $\neg$ and $\wedge$; (d) brackets: (,); and (e) the quantifier symbol $\exists$.

We assume that the universal quantifier and other connectives can be defined in terms of the vocabulary introduced above.

In order to explicate the concept of compositionality we define the syntax and semantics of PL as follows:

(1) Syntax of formulas:

a. If R is an n-ary relation symbol, and $x_1, \ldots, x_n$ are variables, then $f(R, x_1, \ldots, x_n) := R(x_1, \ldots, x_n)$ is a formula.
b. If α is a formula, then so is $f_\neg(\alpha) := \neg\alpha$.
c. If α and β are formulas, then so is $f_\wedge(\alpha, \beta) := (\alpha \wedge \beta)$.
d. If α is a formula, and x_i is a variable, then $f_\exists(x_i, \alpha) := \exists x_i \alpha$ is a formula.

Let $[\![\alpha]\!]^{M,g}$ be the denotation of α relative to the model M and the assignment function g. If α is a formula, $[\![\alpha]\!]^{M,g}$ is always 0 or 1.

(2) Semantics of formulas: Let I be an appropriate interpretation function for constants in a model M, let g and g' be two assignment functions. $g' \sim_i g$ holds if and only if g and g' differ at most in the value they assign to x_i. We define the function $[\![.]\!]$ from formulas into the set of truth values $\{0, 1\}$ recursively as follows:

a. $[\![R(x_1, \ldots, x_n)]\!]^{M,g} = 1$ iff $\langle g(x_1), \ldots, g(x_n)\rangle \in I(R)$
b. $[\![\neg\alpha]\!]^{M,g} = 1$ iff $[\![\alpha]\!]^{M,g} = 0$
c. $[\![(\alpha \wedge \beta)]\!]^{M,g} = 1$ iff $[\![\alpha]\!]^{M,g} = [\![\beta]\!]^{M,g} = 1$
d. $[\![\exists x_i \alpha]\!]^{M,g} = 1$ iff there is a g' with $g' \sim_i g$ and $[\![\alpha]\!]^{M,g'} = 1$

For this formulation to be compositional, the semantics associated with each syntactic rule would have to be some function of the denotation of the parts. Let us assume, as is standard in a PL with terms, that $[\![x_i]\!]^{M,g} = g(x_i)$ and $[\![P]\!]^{M,g} = I(P)$.

(3) **Definition of compositionality**:
A meaning assignment function $[\![\cdot]\!]$ is compositional iff for every structure-building function f there is a semantic operation $\mathcal{O}_f$ such that $[\![f(\gamma_1, \ldots, \gamma_n)]\!] = \mathcal{O}_f([\![\gamma_1]\!], \ldots, [\![\gamma_n]\!])$.

It is well known that there can be no such operation for (2-d). To see this, consider a model with $D = \{a, b, c\}$, $I(R) = \emptyset$, $I(Q) = \{b\}$ and an assignment function g with $g(x) = a$. Relative to this model and this assignment function, it can easily be seen that $[\![R(x)]\!]^{M,g} = 0$. If the meaning assignment were compositional, then $[\![f_\exists(x, R(x))]\!]^{M,g}$ would have to be the result of applying a semantic function $\mathbf{f}_\exists$ to $[\![x]\!]^{M,g} = a$ and $[\![R(x)]\!]^{M,g} = 0$, i.e. $[\![\exists x R(x)]\!]^{M,g} = \mathbf{f}_\exists(a, 0) = 0$. Similarly, we can see that $[\![Q(x)]\!]^{M,g} = 0$ and by compositionality $[\![f_\exists(x, Q(x))]\!]^{M,g}$ would have to be the result of applying the same semantic function $\mathbf{f}_\exists$ to $[\![x]\!]^{M,g} = a$ and $[\![Q(x)]\!]^{M,g} = 0$, i.e. $[\![\exists x Q(x)]\!]^{M,g} = \mathbf{f}_\exists(a, 0) = 1$. But clearly, no function exists with $\mathbf{f}_\exists(a, 0) = 0$ and $\mathbf{f}_\exists(a, 0) = 1$, and therefore we found a model and an assignment function for which the interpretation of existential quantification as stated in (2-d) is not compositional.

1.3 Compositionality Regained: Assignment Functions and the Loss of Alphabetical Innocence

The solution to the compositionality problem is well-known: Instead of assuming that formulas denote truth-values relative to a model and an assignment function, assume that they denote *sets* of assignment functions relative to a model—called global extensions above. So if G is the set of all assignment functions, then global extensions for constants and variables are defined as follows:

(4) Semantics of terms:

a. $[\![x_i]\!]^M$ is that function f from G into D such that $f(g) = g(x_i)$
b. $[\![c_i]\!]^M$ is that function h from G into D such that $h(g) = I(c_i)$

In order to highlight the parallel and the difference between variables and constants, we now added (4-b) to the definition; observe that the interpretation of constants is independent of variable assignments g, and variables are independent of interpretation functions, as is standard in PL. We use t_i as variables for terms, i.e. variables and constants.

(5) Semantics of formulas as sets of assignment functions:

a. $[\![R(t_1, \ldots, t_n)]\!]^M = \{g : \langle [\![t_1]\!]^M(g), \ldots, [\![t_n]\!]^M(g)\rangle \in I(R)\}$
b. $[\![\neg\alpha]\!]^M = G \backslash [\![\alpha]\!]^M$
c. $[\![(\alpha \wedge \beta)]\!]^M = [\![\alpha]\!]^M \cap [\![\beta]\!]^M$
d. $[\![\exists x_i \alpha]\!]^M = [\![\exists]\!]([\![x_i]\!])([\![\alpha]\!]^M) = \{g' :$ there is a g such that $g' \sim_{x_i} g$ and $g \in [\![\alpha]\!]^M\}$

Algebraically speaking, $[\![\exists]\!]$ is the cylindrification of the relation $[\![\alpha]\!]$ at the positions of x_i cf. [4]. As discussed also by [2] p. 10ff such a semantics is compositional. This might be somewhat surprising, since the semantics of quantification mentions the condition $g' \sim_{x_i} g$ and thus seems to refer to a syntactic condition rather than to a purely semantic one. However, since assignments are part of the ontology and since assignments are functions that take variables as their arguments, variables themselves also are part of the ontology, hence semantic objects. As discussed in [14] p. 242 this leads to a certain sloppyness of formalization since variables occur both in the metalanguage of the ontology and in the formulas of PL. This can be avoided by using counterparts, but in the present paper we simply assume that a variable itself is used as its own semantic value, or conversely that semantic objects can enter into syntactic formulas.

Compositionality, however, has been bought at a price. It can easily be seen that in the model specified above with $I(Q) = \{b\}$, $[\![Q(x)]\!]^M \neq [\![Q(y)]\!]^M$, since all assignment functions g with $g(x) = b$ and $g(y) = c$ are in $[\![Q(x)]\!]^M$ but none of them is in $[\![Q(y)]\!]^M$. The price to be paid is loss of alphabetical innocence.

The intuitive idea behind alphabetical innocence, i.e. that the meaning of a formula φ should not change if we replace a free variable in φ by a variable which remains free in φ, can be made more precise by saying that two formulas φ and ψ are alphabetical variants iff there is a bijective function s from the set of free variables in φ into the set of free variables in ψ such that ψ is the result of replacing every free x in φ by $s(x)$. Then we can say (cf. [8], p. 228) that a meaning assignment function is alphabetically innocent iff for every pair φ and ψ of formulas which are alphabetical variants it holds that φ and ψ are synonymous. An equivalent formulation is (6):

(6) **Definition of alphabetical innocence**:

A meaning assignment function $[\![\cdot]\!]$ is alphabetically innocent iff for all formulas $\varphi[x_{a_1}, \ldots, x_{a_n}]$ where $x_{a_1}, \ldots, x_{a_n}$ are the free variables in φ the following holds: if $x_{b_1}, \ldots, x_{b_n}$ are variables such that $x_{b_i} = x_{b_j}$ iff $x_{a_i} = x_{a_j}$ (for all $1 \leq i, j \leq n$) and if $\varphi[x_{b_1}, \ldots, x_{b_n}]$ results from the the replacement of x_{a_i} by x_{b_i} in φ, and all occurences of x_{b_i} are free in $\varphi[x_{b_1}, \ldots, x_{b_n}]$, then $[\![\varphi[x_{a_1}, \ldots, x_{a_n}]]\!] = [\![\varphi[x_{b_1}, \ldots, x_{b_n}]]\!]$.

As shown above, the standard compositional semantics for PL is not alphabetically innocent.

1.4 Fine on Alphabetic Innocence

Fine's crucial step towards solving the antinomy of the variable is making the distinction between intrinsic (or non-relational) and extrinsic (or relational) semantic features. The intrinsic semantic features of two variables x and y are in effect given by the specification of their range, so that the intrinsic features of two variables are identical if their range is identical. However, the specification of the range of a variable does not fully specify its behavior. In particular, the values that the two pairs of variables x, x and x, y can take are not identical, despite the identical range of x and y. This difference in the semantic behavior of x, x and x, y is thus due, not to a difference in the intrinsic features of x and y, but to a semantic feature which is extrinsic to the specification of the range of the individual variable: identical variables cannot simultaneously assume different values, whereas distinct variables can.

If we equate the notion of "semantic role" in the above quote with the notion of extrinsic semantic feature, we can maintain that the semantic roles of x, x and x, y are different, although the semantic roles (i.e. intrinsic semantic features) of x and y taken individually are the same, because the extrinsic semantic feature of the pair of variables x, y is not determined only by the intrinsic semantic features of the individual variables, but also by the semantic relationship between these variables, which implies that x and y can simultaneously take different values. We must thus "recognize that there may be *irreducible* semantic relationships, ones not reducible to the intrinsic semantic features of the expressions between which they hold." [2, 3]

Fine then argues that none of the three approaches to the semantics of PL (the Tarskian, the instantial, and the algebraic approach) provide a philosophically satisfactory solution to the antinomy of the variable. Fine therefore proposes to give up the idea that the role of semantics is to assign a semantic value to each (meaningful) expression of a language, and to embrace instead semantic relationism, according to which the aim of semantics is to "assign a semantic connection to each sequence of expressions", where semantic connections "are intended to encapsulate not only the semantic features of each individual expression but also the semantic relationships between them." [2], p. 25 More precisely, his semantics will assign semantic connections to *coordinated* sequences of expressions, which are pairs consisting of a sequence of expressions and a coordination scheme, which stipulates which occurences of variables are coordinated. The crucial features of Fine's proposal, then are (i) the introduction of coordination schemes, (ii) the assumption that (coordinated) sequences of variables are assigned semantic connections, and (iii) the assumption that the job of semantics is to assign a semantic connection to (coordinated) sequences of expressions.

1.5 Outline of the Theory

But is the move towards a relational semantics in its full generality really forced upon us? We claim that at least for the purposes of solving the antinomy of the variable it suffices to assign *semantic values to sequences of variables*, and to assume coordination schemes. The aim of this paper is to substantiate this claim by designing an explicit formal system, i.e. by modifying the syntax and semantics of PL in such way that the semantics assigns values to (coordinated) expressions, not sequences of formulas, in a compositional and alphabetically innocent way. The modifications of the syntax and semantics of PL presented here do not affect the assumptions that (i) the language contains infinitely many variable symbols, and (ii) that n-ary relation symbols are interpreted as subsets of D^n. This differs from the alphabetically innocent and compositional (variant of) PL presented in [8], which assumes that (i) the set of variables is finite, and (ii) that relation symbols are interpreted by means of so-called concepts, where concepts are sets of relations closed under certain operations on relations.[2]

The basic idea to be pursued in this paper is that sequences of variables are meaningful expressions, and that the meaning assigned to them is not determined by the meaning of the individual variables (i.e. the range of values for the indi-

[2]The motivation for assuming a finite set of variable symbols is the proof, presented in [8], that if relation symbols denote concepts, there are models for which there is no alphabetically innocent and compositional context-free grammar for the language of predicate logic.

vidual variables which has been called the intrinsic semantic feature by Fine), but by the relationship between these variables (their extrinsic semantic feature). We thereby follow Fine in assuming some sort of relationalism embodied in coordination schemes that will be encoded by so-called ϵ/ν-structures (defined in 2.1) which abstract away from the individual shape of a symbol but merely register whether or not two symbols in a sequence are pairwise identical or different. The purpose of this is to provide for a semantic entity that restricts the interpretations of open formulas in such a way that $P(x, y)$ is interpreted differently from $P(y, y)$ because the sequences xy and yy have different semantic denotations, i.e. different ϵ/ν-structures. On the other hand, the sequences xy and yx or zy are "t-equivalent", a notion also defined in 2.1. All this is fully in line with Fine's discussion, except that Fine did not develop a formal system that captures these intuitions.

We will see that, in consequence, ϵ/ν-structures play a role similar to assignment functions: both assignment functions and ϵ/ν-structures are considered semantic entities that guarantee a compositional treatment of PL. But whereas the latter involve symbols for variables in their domain and thus enhance the ontology with syntactic objects taken from the language of PL, the objects represented by ϵ/ν-structures only reflect (syntactic) properties of the semantic relations already in the model; their only function is to identify positions of such relations.

Composing simple formulas into complex ones involves two crucial conditions. The first concerns conjunction. Here we have to specifying which variables of the subformulas to be conjoined are interpreted as the same symbols in the entire complex formula. If alphabetical variants are to be synonymous, then the actual names of the variables determine which variables stand for the same individuals within a sequence of variables, but not across two sequences. For example, formulas like $P(x, y)$ and $Q(x, x)$ clearly express whether or not their arguments differ; however, when combining the two we must keep in mind that $Q(x, x)$ is equivalent to $Q(y, y)$ and $Q(z, z)$ by alphabetic innocence, hence we cannot be sure whether the result of conjunction should be $(P(x, y) \wedge Q(x, x))$ or $(P(x, y) \wedge Q(y, y))$ or $(P(x, y) \wedge Q(z, z))$. At this point we have to disambiguate what we call an ambiguous merge (to be defined in Section 2.2). Such a disambiguation will state explicitly which variables stand for the same individual across two sequences of subformulas. If a disambiguation states that the variable in the i-th position of a sequence of variables σ_1 is identical with the variable in the j-th position of σ_2, we shall say that the two variables are coordinated. In the above example, a coordination of the second variable in $P(x, y)$ with the first (and second) variable of $Q(x, x)$ will yield $(P(x, y) \wedge Q(y, y))$ as coordinated formula.

The second crucial condition is quantification. There are basically two possibilities: one is to reduce the arity of a quantified formula by letting the existential quantification rule delete all occurrences of the variable that gets bound, so that all variables that occur in the resulting sequence can be guaranteed to be variables not yet bound by a quantifier. Compositionality now requires the semantic arity of a formula to be reduced accordingly. This arity-reducing syntax and semantics is worked out in Sections 3 and 4, in which we prove the equivalence between the logic developed in this paper and classical predicate logic.

Superficially, a non-reducing second variant that does not delete bound variables looks simpler, because we could dispense with all arity-reducing operations. Hence, bound variables could still be part of the syntactic as well as the semantic representation, they represent an "internal semantic feature" as their denotation corresponds to what is called the cylindrification of the argument position they represent, as will be discussed in Section 5. We will show that such a simple semantics is feasable, but will not allow for an equally simple translation procedure into ordinary PL. Simplicity can be regained, however, if we make a relevant syntactic distinction between free and bound variables. Compositionality then requires that an analogous distinction is also made in the semantics of variables, which somewhat complicates the system and thereby compensates for the lack of arity reduction.

In Section 6, we show how to add constants and functions to the system. As an application of the theory we will finally demonstrate how the system can solve a problem in Montague Grammar.

2 Basic Notions

2.1 Same or Different: ϵ/ν-Structures

Let z be a word over an alphabet K, i.e. a concatenation of symbols. z has length n iff $z \in K^n$. We will use variables $x, y, x_1, \ldots, x_n$ to denote occurrences of symbols in a word. If, for example, K is the alphabet of English and z = *pop*, then z is a sequence of symbols $x_1x_2x_3$ such that $x_1 = x_3$, which says that the first symbol of the word is identical to the third.

We will define a function that abstracts away from the individual properties of words and the shape of the symbols while at the same time recording whether or not two entities in the sequence of symbols in a word are identical. This is the only information about z provided by the function δ defined as follows.

(7) **Definition of δ:**

Let $z = x_1 \ldots x_n$. Define $\delta(\langle i, j\rangle, z)$ as that function that assigns to a pair of positions $\langle i, j\rangle$, $1 \leq i, j \leq n$, in z the pair $\langle\langle i, j\rangle, \epsilon\rangle$ if $x_i = x_j$, and $\langle\langle i, j\rangle, \nu\rangle$ otherwise.

Given a word z, the function δ is uniquely determined by z.

The symbol ϵ means "equal", ν is "non-equal". Of course, any other symbols $(+, -; 0, 1)$ could do the job; here and in the following definition we follow the notation and the terminology in [9], including the following definition of so-called ϵ/ν-structures:

(8) **Definition of ϵ/ν-structures:**

κ is an ϵ/ν-structure of rank n iff κ is a function from all pairs of integers i, j with $1 \leq j \leq i \leq n$ into the set $\{\epsilon, \nu\}$ such that for all i, j, k:

a. $\kappa(i, i) = \epsilon$,
b. if $\kappa(i, j) = \epsilon$ and $\kappa(j, k) = \epsilon$, then $\kappa(i, k) = \epsilon$.

To illustrate, a word like *banana* will be assigned the ϵ/ν-structure κ in (9):

(9) $$\begin{aligned}\kappa = \{ & \langle\langle 1,1\rangle,\epsilon\rangle,\ \langle\langle 2,1\rangle,\nu\rangle,\ \langle\langle 3,1\rangle,\nu\rangle,\ \langle\langle 4,1\rangle,\nu\rangle,\ \langle\langle 5,1\rangle,\nu\rangle,\ \langle\langle 6,1\rangle,\nu\rangle,\\ & \langle\langle 2,2\rangle,\epsilon\rangle,\ \langle\langle 3,2\rangle,\nu\rangle,\ \langle\langle 4,2\rangle,\epsilon\rangle,\ \langle\langle 5,2\rangle,\nu\rangle,\ \langle\langle 6,2\rangle,\epsilon\rangle,\\ & \langle\langle 3,3\rangle,\epsilon\rangle,\ \langle\langle 4,3\rangle,\nu\rangle,\ \langle\langle 5,3\rangle,\epsilon\rangle,\ \langle\langle 6,3\rangle,\nu\rangle,\\ & \langle\langle 4,4\rangle,\epsilon\rangle,\ \langle\langle 5,4\rangle,\nu\rangle,\ \langle\langle 6,4\rangle,\epsilon\rangle,\\ & \langle\langle 5,5\rangle,\epsilon\rangle\ \ \langle\langle 6,5\rangle,\nu\rangle,\\ & \langle\langle 6,6\rangle,\epsilon\rangle,\ \}\end{aligned}$$

And the same structure will be assigned to the word *Rococo*.

Assume now that K consists of the variables of a predicate logic and that the sequences of variables like those in $R(y_1,\dots y_n)$ form a word $y_1\dots y_n$. ϵ/ν-structures will then serve as the denotion of $y_1\dots y_n$ as defined in (10):

(10) **Denotation of σ:**
If σ is a word of length n, then the denotation of σ, written as $[\![\sigma]\!]$, is an ϵ/ν-structure of rank n such that $[\![\sigma]\!] := \{\langle\langle i,j\rangle,\alpha\rangle : \langle\langle i,j\rangle,\alpha\rangle = \delta(\langle\langle i,j\rangle,\sigma\rangle)$ and$1 \leq j \leq i \leq n\}$.

The reader should verify that for any pair of positions i, j of σ with $i \geq j$, (i) the pair $\langle\langle i,j\rangle,\epsilon\rangle$ is in $[\![\sigma]\!]$ if and only if the variable at position i in σ is identical with the variable at position j in σ, and (ii) the pair $\langle\langle i,j\rangle,\nu\rangle$ is in $[\![\sigma]\!]$ iff the variables at positions i and j are not identical.

The role of such denotations in the logic to be developed is to have a semantic counterpart to syntactic sequences of variables that help to identify variables and coordinate them. Clearly, identical variables constrain interpretations in the way described in the introduction, in that identical variables in a sequence as in $Ryxy$ will have the effect of restricting the interpretation of R to only those sequences in $I(R)$ that have identical values in the first and the third position of $I(R)$. In order to formalize this intuition, we define the following auxiliary notion.

(11) **Definition of "t conforms to κ":**
An n-tuple $t = \langle a_1,\dots,a_n\rangle$ conforms to an ϵ/ν-structure κ of rank n iff

$$\forall i\forall j[1 \leq i, j \leq n \rightarrow (\kappa(i,j) = \epsilon \rightarrow a_i = a_j)]$$

A first application of (11) is straightforward: Assume that $[\![\sigma]\!]$ is an ϵ/ν-structure of rank n, where σ is a string of variables $x_1\dots x_n$. Let $R\sigma$ be an atomic formula and I an interpretation function for R in a model. Now assume that the denotation of $R\sigma$ is $\{s \in I(R) : s$ conforms to $[\![\sigma]\!]\}$. It now follows that the denotation of $Rxyz$ is the same as that of $Ruvw$, namely $I(R)$ itself, whereas the denotation of $Rxxy$ picks out only those 3-tupels in the relation that have identical first and second positions.

The next step, then, will be a semantic account of logical connectives and of quantification, alongside with a definition of truth and satisfaction.

Before going on a couple of remarks seem to be in order. First, it may seem that the denotations of sequences could be simplified by formulating them as sets of equivalence classes of positions (for the example above $\kappa = \{\{1\},\{2,4,6\},\{3,5\}\}$). But as we shall see below, if one wants to express the difference between free and bound variable one would need a more complicated approach than denotations as sets of equivalence classes; in fact, it would be necessary to consider two sets of equivalence classes, one for free and one for bound variables. And of course one would have to relate the two classes to each other. A much simpler approach would still use ϵ/ν-structures, but with a slight modification: in addition to pairs of the form $\langle\langle i, i\rangle, \epsilon\rangle$ for free positions, we can introduce pairs of the form $\langle\langle i, i\rangle, \nu\rangle$ for bound positions, as defined in Section 5. This has the slight technical advantage that the position of the different types of variables is still determined in a single string of objects.

Second, it should be noted that the semantic effect expressed by ϵ/ν-structures can also be captured syntactically by defining a syntactic notion (called "trito-structure equivalence" in [9]) which mirrors the equivalence of ϵ/ν-structures:

(12) **Definition of t-equivalence:**
Two words z_1 and z_2 of length n are t-equivalent iff for all positions i, j with $1 \leq i, j \leq n$: $\delta(\langle i, j\rangle, z_1) = \delta(\langle i, j\rangle, z_2)$. If z_1 and z_2 are t-equivalent, we write $z_1 =_t z_2$.

It follows that all symbols as words of lenght 1 are t-equivalent. xy is t-equivalent with yx and xz, but not with xx or bb, the latter two being t-equivalent again. In general, we have the following theorem:

(13) $\sigma =_t \sigma'$ iff $[\![\sigma]\!] = [\![\sigma']\!]$

Third, it should be noted that ϵ/ν-structures, though serving as the denotation of sequences of variables, will not appear directly as the denotation of formulas in the language we will define below: here we only see relations in a given domain D. However, they are crucial in determining these relations, hence still belong to the compositional semantics, as we will see immediately.

2.2 Coordination and Ambiguous Merge

Let us now look at conjunction. Assume we want to "coordinate" Pxy and Qzx. What is the result? We know what the two ϵ/ν-structures look like, namely $[\![xy]\!] = [\![zx]\!] = \{\langle\langle 1, 1\rangle, \epsilon\rangle, \langle\langle 2, 1\rangle, \nu\rangle, \langle\langle 2, 2\rangle, \epsilon\rangle\}$. But since $[\![xy]\!]$ and $[\![zx]\!]$ abstract away form the actual symbols, we cannot infer, as the traditional notation would suggest, that we have to identify the first position of P with the second position of Q. This kind of information is not yet available, so clearly some kind of additional information is required to get a result. This kind of information is, we believe, related to what [2] calls a *coordination scheme* in the following quote (p. 30):[3]

> In the first place, the syntactic object of evaluation will no longer be a sequence of expressions but a *coordinated* sequence of expressions. This is a sequence of expressions $E_1, E_2, \ldots, E_n$ along with a *coordination scheme* $\mathcal{C}$ which tells us when two free occurrences of the same variable are to be coordinated (formally,

[3]Note that according to Fine the coordination scheme only tells us which occurrences of the *same* variable are coordinated. As will be seen below, our notion of *disambiguation* may also coordinate occurrences of different variables.

a coordination scheme is an equivalence relation on the free occurrences of variables in the sequence subject to the requirement that it only relate occurrences of the same variable.) [emphasis in the original]

We will first define an "ambiguous merge" of two words in such a way that the result corresponds to all possible coordination schemes, which will now be represented as embeddings of two ϵ/ν-structures into one. Each such larger structure will be called a disambiguation. More precisely, we will first define the ambiguous merge of two strings of symbols, the result being a set of sequences each of which has an ϵ/ν-structure that conforms to the internal structure of its components.

As always, sequences of symbols are generated by concatenation; a sequence of length n results from concatenating a symbol to a sequence of length $n - 1$. The concatention of two words σ_1 and σ_2 is denoted by $\sigma_1\sigma_2$. A word of length n in K^n is also called an n-ary K-sequence.

We now define the range of possible desambiguations (all the possibilities of identifying the positions of two sequences) as ambiguous merge of sequences of variables:

(14) **Definition of ambiguous merge:**
Let σ_1 and σ_2 be K-sequences. Then the ambiguous merge $\sigma_1@\sigma_2$ is the set of all K-sequences $\sigma_1'\sigma_2'$ such that $\sigma_1' =_t \sigma_1$ and $\sigma_2' =_t \sigma_2$.

Any illustration of the above definition faces the difficulty that for each σ there are infinitely many t-equivalent sequences σ'. We therefore find it convenient to define equivalence classes or isomorphic representations of K-sequences. One way of doing so is to assume a total ordering W on K, referred to as a sequence of distinct elements in (16). We thus adopt the following conventions:

(15) **Definition of *i*-th projection:**
If σ is a word of lenght n, let $\pi_i(\sigma)$ be the i-th symbol in σ; if s is an n-tupel, let $\pi_i(s)$ be the i-th element of s; these elements are called the i-th projection of $s/sigma$.

(16) **Definition of normalized sequence:**
Let W be an infinite sequence in $K*$ such that for all $i \neq j, \pi_i(W) \neq \pi_j(W)$. A K-sequence $\sigma \in K^n$ is normalized (with respect to W) iff the following holds:

$$\text{if } \pi_i(\sigma) = \pi_j(\mathrm{W}), n \geq j > 1, \text{ then } \pi_{j-1}(\mathrm{W}) = \pi_k(\sigma) \text{ for some } k < i.$$

Assuming W as given in (17), the sequences in (18) are all normalized sequences of respective lengths.

(17) W = ⟨○, □, ◇, ▽, △ . . .⟩
(18) length 0: ⟨⟩ (= Λ)
length 1: ○
length 2: ○○, ○□
length 3: ○○○, ○○□, ○□○, ○□□, ○□◇

length 4: ○○○○, ○○○□, ○○□○, ○○□□, ○○□◇, ○□○○, ○□○□,
○□○◇, ○□□○, ○□□□, ○□□◇, ○□◇○, ○□◇□, ○□◇◇, ○□◇▽.
length 5: etc.

Returning again to the concatenation of K-sequences, an example is given in (19). There are seven equivalence classes of disambiguations of ○□@○□, which are represented by normalized K-sequences. The third column indicates the positions identified, and the fourth column contains an example out of the infinitely many K-sequences contained in one equivalence class:

(19)

equiv. class	normalized K-seq.	positions identified	example
1	○□◇▽	–	△◇▽□
2	○□○◇	1-3	△□△▽
3	○□◇○	1-4	□▽△□
4	○□□◇	2-3	◇□□△
5	○□◇□	2-4	▽◇○◇
6	○□○□	1-3, 2-4	△○△○
7	○□□○	1-4, 2-3	◇○○◇

Taking up Fine's notion of Coordination Scheme, the positions identified are those that are "coordinated" by a "disambiguation". We thus define:

(20) **Definition of coordination:**
Assume that $\sigma_1 \in K^m, \sigma_2 \in K^n$ and that $\sigma_1'\sigma_2'$ is a disambiguation of $\sigma_1 @ \sigma_2$. $\sigma_1'\sigma_2'$ coordinates a position i in σ_1 with a position j in σ_2 iff $\pi_i(\sigma_1'\sigma_2') = \pi_{m+j}(\sigma_1'\sigma_2')$.

We emphasize that a normalized K-sequence (also called a Kenogramm in [9]) is just a convenient way of representing equivalence classes of K-sequences; normalized K-sequences will be used in informal expositions below, but for the sake of alphabetical innocence, we will **not** make use of them in the formal definitions to come.[4]

Identification of positions as illustrated in (20) will also play a major role in binding. Consider a formula like $P(x, y, x)$ in normal notation, or $\langle P, ○□○\rangle$ in ours. Adding a quantifier and a variable requires an alphabetically innocent identification of the variable to be bound. This will be achieved by merging a symbol with the sequence ○□○, as in ○@○□○. Disambiguation leads to three different outcomes. Either the disambiguation γ targets the first (or the third) position in σ = ○□○, which means that γ can be represented as ○○□○. Alternatively,

[4]Ede Zimmermann proposed to us that it would be much simpler to take the normalized sequences themselves as the denotations of sequences of variables, rather than ϵ/ν-structures. In particular, we could take W to be the set of natural numbers. At this point the objection is correct, but when it comes to the distinction between free and bound variables we do not see how to calculate with such sequences in a straightforward and natural way. Cf. also the remark above that the opposition between free and bound would necessitate the introduction of two (intertwined) equivalence classes.

it may target the second position, with γ corresponding to $\bigcirc\Box\bigcirc\Box$. Finally, vacuous quantification is represented by $\bigcirc\Box\Diamond\Box$.

3 Arity Reducing Predicate Logic (ARPL)

The following proposal has in common with the proposal in [13] that existential quantification reduces the arity of a relation, so that in effect a position cannot be bound twice. The main difference, however, concerns the status of variables. Quine's aim is to show that (sequences of) variables can be dispensed with in the formulation of PL. Our aim, on the other hand, is to provide an alphabetically innocent formalisation of PL **with** variables.

We therefore have to define two operations: one that kicks out variables from a K-sequence; and one that correspondingly reduces the arity of its denotation, i.e. the corresponding ϵ/ν-structure. In both cases the variable to be bound, e.g. by $\exists x_i$ in a formula $P(x_1, \ldots, x_i, \ldots, x_n)$, is represented in a K-sequence as the first variable in the sequence $x_i, x_1, \ldots, x_i, \ldots, x_n$, so it suffices to compare elements of a sequence to its first element. Let us start with the syntactic reduction, called 1-reduction, that kicks out a symbol:

(21) **Definition of 1-reduction**:

Let $\sigma := x_1, \ldots, x_n$ be a K-sequence with $n \geq 1$, Λ the empty string, and $+$ concatenation. Then the 1-reduction $r_1(\sigma)$ is recursively defined as follows:

$$r_1(x_1, \ldots, x_n) = \begin{cases} \Lambda, & \text{if } n = 1 \\ r_1(x_1, \ldots, x_{n-1}), & \text{if } n > 1 \text{ and } x_1 = x_n \\ r_1(x_1, \ldots, x_{n-1}) + x_n, & \text{if } n > 1 \text{ and } x_1 \neq x_n \end{cases}$$

For example,
$$\begin{aligned} r_1(\bigcirc\Box\bigcirc\Diamond) &= r_1(\bigcirc\Box\bigcirc)\Diamond && \text{(third clause)} \\ &= r_1(\bigcirc\Box)\Diamond && \text{(second clause)} \\ &= r_1(\bigcirc)\Box\Diamond && \text{(third clause)} \\ &= \Box\Diamond && \text{(first clause)} \end{aligned}$$

Suppose σ is $xyxz$. The first variable will correspond to the variable we find in $\exists x$ in traditional notation. The variables in the remainder of the formula are yxz. The remaining free variables should be yz. As the reader may easily verify, $r_1(xyxz) = yz$.

Next, let us define the sets Fml_n of formulas of rank n:

(22) **Syntax of ARPL:**

a. Let R be a predicate constant of arity n and $\sigma \in K^n$. Then $f(R, \sigma) = \langle R, \sigma \rangle \in Fml_n$

b. If $\varphi = \langle \alpha, \sigma_1 \rangle \in Fml_n$, $\psi = \langle \beta, \sigma_2 \rangle \in Fml_m$, and $\sigma \in \sigma_1 @ \sigma_2$, then
$$f_\wedge(\varphi, \psi, \sigma) = \langle (\varphi \wedge \psi), \sigma \rangle \in Fml_{n+m}[5]$$

[5] Alternatively, we could define $f_\wedge$ by
$$f_\wedge(\varphi, \psi, \sigma) = \langle (\alpha \wedge \beta), \sigma \rangle \in Fml_{n+m}$$
This leaves the semantics unaffected, but we would lose unique readability, ie. the unambiguous decomposition of a conjunction into its constituents.

c. If $\langle \alpha, \sigma \rangle \in Fml_n$, then
$$f_{\neg}(\langle \alpha, \sigma \rangle) = \langle \neg\alpha, \sigma \rangle \in Fml_n$$
d. If $\varphi = \langle \alpha, \sigma \rangle \in Fml_n$, $k \in K^1$, $\sigma' \in k@\sigma$, and $\sigma'' =_t r_1(\sigma')$, then
$$f_{\exists}(\varphi, \sigma') = \langle \exists\sigma'\varphi, \sigma'' \rangle \in Fml_m$$
where m is the arity of σ''

Here is a complex example with two quantifiers. Consider the PL-formula (23-a), equivalent to (23-b), which can be interpreted as saying that there are infinitely many prime numbers (with PN = prime number and $<(x, y)$ as the PL-equivalent to $x < y$):

(23) a. $\forall x(PN(x) \rightarrow \exists y(PN(y) \wedge <(x, y)))$
b. $\neg\exists x(PN(x) \wedge \neg\exists y(PN(y) \wedge <(x, y)))$

Translating (23-b) into ARPL in a way that only uses normalized sequences in the order $\langle \bigcirc, \Box, \ldots \rangle$, we start with $\langle PN, \bigcirc \rangle$ and $\langle <, \bigcirc\Box \rangle$, conjoining them with a desambiguation that actually twiddles the argument symbols of the relation $<$:

(24) $\langle (\langle PN, \bigcirc \rangle \wedge \langle <, \bigcirc\Box \rangle), \bigcirc\Box\bigcirc \rangle$

Quantifying over the second position of the relation $<$ amounts to targeting the third position of the complex relation and thus calls for $\bigcirc\bigcirc\Box\bigcirc$ as an appropriate desambiguation for the quantifying expression. As the r$_1$-reduction of $\bigcirc\bigcirc\Box\bigcirc$ only leaves the symbol $\Box$, this is normalized to $\bigcirc$, which then yields:

(25) $\langle \exists\bigcirc\bigcirc\Box\bigcirc\langle (\langle PN, \bigcirc \rangle \wedge \langle <, \bigcirc\Box \rangle), \bigcirc\Box\bigcirc \rangle, \bigcirc \rangle$

Continuing with negation and the remainder of the formula we finally get:

(26) $\langle \neg\exists\,\bigcirc\bigcirc\bigcirc\langle (\langle PN, \bigcirc \rangle \wedge \langle \neg\exists\bigcirc\bigcirc\Box\bigcirc\langle (\langle PN, \bigcirc \rangle \wedge \langle <, \bigcirc\Box \rangle), \bigcirc\Box\bigcirc \rangle, \bigcirc \rangle), \bigcirc\bigcirc \rangle, \Lambda \rangle$

In semantics, all formulas will denote n-tuples, where n is the number of free variables in it. Existential quantification will reduce the arity of a relation. This semantic reduction of a relation eliminates (or collapses) all projections which are bound by the existential quantifier. We therefore have to define an operation akin to r_1-reduction that operates on the semantic denotations of K-sequences; the effect is exactly the same, namely elimination of positions that are bound according to the disambiguation by sameness with its first position. This first position is an auxiliary semantic construct corresponding to the variable in $\exists x$, which will become clear from (31-f) below.

(27) **Definition of κ-reduction**:
Let s be a sequence of length i and κ a constant ϵ/ν-structure of arity $n \geq i$. Then:

$$r_\kappa(s) = \begin{cases} \emptyset & \text{if } i = 1 \\ r_\kappa(\langle \pi_1(s), \ldots, \pi_{i-1}(s) \rangle), & \text{if } i > 1 \text{ and } \kappa(1, i) = \epsilon \\ r_\kappa(\langle \pi_1(s), \ldots, \pi_{i-1}(s) \rangle) + \pi_i(s), & \text{if } i > 1 \text{ and } \kappa(1, i) = \nu \end{cases}$$

Note in passing that the meaning of + as applied to n-tuples should be defined:

(28) If $s = \langle t_1, \ldots t_n \rangle$ and $s' = \langle t'_1, \ldots t'_m \rangle$, then $s + s' = \langle t_1, \ldots t_n, t'_1, \ldots t'_m \rangle$.

Before we can state the semantics of ARPL, we have to slightly generalize the definition of Cartesian products when applied to sequences; this will amount to a product of concatenations:

(29) Let R be an n-place relation, i.e. a set of n-tupels, and Q a set of m-tupels. Then $R \otimes Q$ is the set of all $n+m$-tuples such that if $s \in R \otimes Q$, then $s = s_1 + s_2$ and $s_1 \in R$ and $s_2 \in Q$.

(30) Let σ be a K-sequence of length n, then $[\![\sigma]\!]_D := \{s \in D^n : s$ conforms to $[\![\sigma]\!]\}$

Note that if Λ is the empty sequence, $[\![\Lambda]\!]_D = \{s \in D^0 : s$ conforms to $[\![\Lambda]\!]\} = \{\Lambda\} = \{\emptyset\}$.

(31) **Semantics of ARPL:**
Let I be an interpretation of the relational constants of L.

a. $[\![\sigma]\!]$ as in (10)
b. $[\![R]\!] = I(R)$ for all constants R.
c. $[\![f(R, \sigma)]\!] = \mathcal{O}([\![R]\!], [\![\sigma]\!]) = \langle [\![R]\!], [\![\sigma]\!]_D \rangle$
d. $[\![f_\wedge(\varphi, \psi, \sigma)]\!] = \mathcal{O}_\wedge([\![\varphi]\!], [\![\psi]\!], [\![\sigma]\!]) = \langle \pi_1([\![\varphi]\!]) \otimes \pi_1([\![\psi]\!]), [\![\sigma]\!]_D \rangle$
e. $[\![f_\neg(\varphi, \sigma)]\!] = \mathcal{O}_\neg([\![\varphi]\!], [\![\sigma]\!]) = \langle D^n \setminus \pi_1([\![\langle\varphi, \sigma]\!]), [\![\sigma]\!]_D \rangle$, n = the rank of $[\![\sigma]\!]$
f. $[\![f_\exists(\varphi, \sigma)]\!] = \mathcal{O}_\exists([\![\varphi]\!], [\![\sigma]\!]) = \langle$
$\{t :$ there is an $s \in D \otimes \pi_1([\![\varphi]\!])$ such that $t = r_{[\![\sigma]\!]}(s)$
and if $\langle\langle 1, i\rangle, \epsilon\rangle \in [\![\sigma]\!]$ and $\langle\langle 1, j\rangle, \epsilon\rangle \in [\![\sigma]\!]$ then $\pi_i(s) = \pi_j(s)\}$,
$\{t' :$ there is an $s' \in [\![\sigma]\!]_D : t' = r_{[\![\sigma]\!]}(s'))\}\rangle$

Note that both 1-reduction and κ-reduction cut away the "identifier" of the variable (i.e. the first position of the disambiguation). In (31-f), we "provisionally" add a "dummy" first position to the relational denotation of φ that serves as the identifier in the semantics and that will subsequently be removed by r_κ as a consequence of r_1 and r_κ being defined in a parallel manner.[6] Moreover, the removed entities must conform to the identity of x, i.e. the values for each occurrence of x must be identical, as this information is no more available in the reduced K-sequence.

Other connectives can easily be defined. Given that (31-d) is equivalent to (32-a), we define the usual operators as in (32-b-d):

(32) a. $\mathcal{O}_\wedge([\![\varphi]\!], [\![\psi]\!], [\![\sigma]\!]) = \langle\{\{s_1 s_2 : s_1 \in \pi_1([\![\varphi]\!])$ and $s_2 \in \pi_1([\![\psi]\!])\}, [\![\sigma]\!]_D\rangle$
b. $\mathcal{O}_\vee([\![\varphi]\!], [\![\psi]\!], [\![\sigma]\!]) = \langle\{s_1 s_2 : s_1 \in \pi_1([\![\varphi]\!])$ or $s_2 \in \pi_1([\![\psi]\!]), [\![\sigma]\!]_D\rangle$
c. $\mathcal{O}_\rightarrow([\![\varphi]\!], [\![\psi]\!], [\![\sigma]\!]) = \langle\{s_1 s_2 :$ if $s_1 \in \pi_1([\![\varphi]\!])$ then $s_2 \in \pi_1([\![\psi]\!]), [\![\sigma]\!]_D\rangle$
d. $\mathcal{O}_\leftrightarrow([\![\varphi]\!], [\![\psi]\!], [\![\sigma]\!]) = \langle\{s_1 s_2 : s_1 \in \pi_1([\![\varphi]\!])$ iff $s_2 \in \pi_1([\![\psi]\!]), [\![\sigma]\!]_D\rangle$
e. etc.

[6] In a previous much longer version of this paper we added the identifier without subsequent reduction of the arity. We will not represent such an arity enhancing variant here, as it merely turns out as a notational variant of the arity preserving logic to be discussed in Section 5.

We may also add identity: for any two variables x_1 and x_2, $f_{\equiv}(x_1, x_2) = x_1 \equiv x_2$ and $\mathcal{O}_{\equiv}([\![x_1x_2]\!]) = \langle\{s_1s_2 : s_1 \in D, s_2 \in D \text{ and } s_1 = s_2\}, [\![x_1x_2]\!]_D\rangle$.

Note that as a consequence of (31-f), a zero-place relation (a closed sentence) will denote $\langle\{\emptyset\}, \{\emptyset\}\rangle$ if the zero-place relation (the sentence) is satisfiable, hence true; whereas it denotes $\langle\emptyset, \{\emptyset\}\rangle$ otherwise, hence when false. Note also that $D^n \otimes D^0 = D^n$, $D^n \otimes \emptyset = \emptyset$, $D^0 \setminus \{\emptyset\} = \emptyset$, and $D^0 \setminus \emptyset = \{\emptyset\}$.

(33) **Definition of truth and satisfiability:**
Assume φ is a formula of ARPL and $[\![\varphi]\!] = \langle\rho, \tau\rangle$. Then

a. φ is true iff $\tau \subseteq \rho$.
b. φ is satisfiable iff $\tau \cap \rho \neq \emptyset$.
c. φ is false iff $\tau \cap \rho = \emptyset$.

Let us discuss some trivial examples. Assume R is identity and $\varphi = \langle R, xy\rangle$. Then $[\![\varphi]\!] = \langle I(R), D^2\rangle$. The formula is satisfiable but not true. If $\varphi = \langle R, xx\rangle$, $[\![\varphi]\!] = \langle I(R), \{\langle a, a\rangle : a \in D\}$. Since $\{\langle a, a\rangle : a \in D\} = I(R)$, the formula is true. If $I(R) = D^2$, both formulas are true.

It should be mentioned that the denotations of $(A \wedge B)$ and $(B \wedge A)$ in ARPL are not necessarily identical. This is because the concatenation $[\![\varphi]\!] \otimes [\![\psi]\!]$ is in general different from $[\![\psi]\!] \otimes [\![\varphi]\!]$. However, this does not affect truth or satisfiability: if one relation is satisfiable (true), the other must be as well. The deeper reason for this asymmetry lies in the fact that the representation of any relation in terms of sequences is conventionalized; for example, the same two-place relation can be represented as R and as R^- with different denotations. The semantic roles of a relation are syntactically encoded by the order of their appearance in R or R^- respectively, hence by a syntactic convention that should be independent of its meaning. The same applies to the linear representation of conjunction as either (A and B) or (B and A). There are technical ways to get around this linear effect (cf. [14] p. 80ff or [7]) which could also be applied to the problem at hand, but for the present purpose we will not bother (but see Section 4.2 for further discussion).

Recall that the analogue of (33) for compositional PL is (34):

(34) Any formula φ of PL is

a. true iff $[\![\varphi]\!] = G$, the set of all assignments
b. satisfiable iff $[\![\varphi]\!] \neq \emptyset$, false iff $[\![\varphi]\!] = \emptyset$

We will prove these concepts to be equivalent for PL and ARPL later in Section 4; for the time being let us only look at atomic formulas. Assume that $\varphi = P(x_1, x_2, \ldots, x_n)$ is an atomic formula of PL and $\varphi' = \langle P, \langle x_1x_2 \ldots, x_n\rangle\rangle$ is the corresponding Fml_n in ARPL, where x_i is a meta-variable and PL and ARPL share the same set of variables. Note that the variables denoted by x_i and x_j are not necessarily distinct.

(35) φ is satisfiable iff φ' is.

Proof φ' is satisfiable iff $I(P) \cap [\![x_1x_2 \ldots x_n]\!]_D \neq \emptyset$; iff there is an s, $s \in I(P)$ and s conforms to $[\![x_1x_2 \ldots x_n]\!]$; iff $s \in I(P)$ and there is an assignment g such that $s = \langle g(x_1), g(x_2), \ldots g(x_n)\rangle$; iff for some g, $\langle g(x_1), g(x_2), \ldots g(x_n)\rangle \in I(P)$; iff φ is satisfiable. □

(36) φ is true iff φ' is.

Proof φ is true iff $[\![\varphi]\!] = G$; iff for all g, $\langle g(x_1), g(x_2), \ldots g(x_n)\rangle \in I(P)$; iff for all $s \in D^n$, if s conforms to $[\![x_1 x_2 \ldots x_n]\!]$, then $s \in I(P)$; iff $[\![x_1 x_2 \ldots x_n]\!]_D \subseteq I(P)$; iff φ' is true. □

Clearly, it already holds in PL that the conditions for truth and satisfiability are identical for all alphabetic variants of φ, though their denotations are different, whereas it now follows in ARPL that even their denotations are identical.

It should by now be plausible that we get the intended result, namely a version of PL that is both compositional and radically innocent. We have shown the logical equivalence for atomic formulas above; in order to prove this for all formulas, we will define a syntactic correspondence relation between the formulas of PL and of ARPL and show that for any formula of PL its truth conditions are identical to those of its corresponding formula in ARPL.

Before doing so in the next section, let us briefly come back to the issue of compositionality which in classical PL hinges on the assumption that value assignments are primitive building blocks that need not be analysed any further. Recall that it was only by the help of such functions that compositionality could be achieved. Similarly, ϵ/ν-structures are presupposed in the present framework as primitive; in particular it is assumed that each K-sequence is interpreted by (10). However, this still does not take into account that the sequences can be built up by concatenation of symbols and that likewise ϵ/ν-structures could be composed out of simpler ϵ/ν-structures for shorter K-sequences. The question then arises as to whether it is possible to arive at a compositional semantics for K-sequences; in its present formulation the semantics effectively uses an infinite supply of operations that identify arbitrary positions of a sequence.

Such a semantics is indeed possible along the lines of [13]; we can then get along with only a finite number of elementary syntactic and semantic operations that could build up the syntax and semantics of K-sequences in a recursive way. However, such a semantics would be rather artificial, and as Fine writes on p. 21 of his book, “we thereby loose what is most distinctive about the use of variables”, nor would we gain an understanding of our use of variables. We therefore conclude that, although technically feasable, this aspect of compositionality is not intended to apply to the domain of K-sequences, and that therefore the holistic way we conceive of ϵ/ν-structures as an indication of sameness vs. difference in a relation is all we need on the conceptual level, without there being any need of further recursive analysis.

4 Calculus and Semantic Equivalence

4.1 Translation, Entailment, and Deductions

The notions of entailment and deduction can best be understood by realizing that there is a truth preserving correspondence between the formulas of classical PL and our systems; we will demonstrate this by specifying a translation function from PL to ARPL and back from ARPL to PL.

The most straightforward way to translate from one language into the other is to assume that both languages use the same set of variables. This assumption will guarantee that the translation is unambiguous; other possible procedures would be in need of normalizations which are unnecessary for the simple demonstration of translatability:

(37) Translation from PL into ARPL:

a. $T(P(x_1, x_2, \ldots, x_n)) = \langle P, x_1 x_2 \ldots x_n \rangle$
b. $T((A \wedge B)) = \langle (T(A) \wedge T(B)), \sigma \rangle$ where $\sigma = \pi_2(T(A))\pi_2(T(B))$
c. $T(\neg A) = \langle \neg \pi_1(T(A)), \pi_2(T(A)) \rangle$
d. $T(\exists x A) = \langle \exists \sigma T(A), r_1(\sigma) \rangle$ where $\sigma = x\pi_2(T(A))$

(38) Translation from ARPL into PL:

a. $T^-(\langle P, x_1 x_2 \ldots x_n \rangle) = P(x_1, x_2, \ldots, x_n)$
b. $T^-(\langle (\langle \alpha, \sigma_1 \rangle \wedge \langle \beta, \sigma_2 \rangle), \sigma \rangle) = (T^-(\langle \alpha, \sigma_1' \rangle) \wedge T^-(\langle \beta, \sigma_2' \rangle))$ with $\sigma_1'\sigma_2' = \sigma$, where σ_1 has the same length as σ_1', and σ_2 has the same length as σ_2'
c. $T^-(\langle \neg \alpha, \sigma \rangle) = \neg T^-(\langle \alpha, \sigma \rangle)$
d. $T^-(\exists \sigma' \varphi, r_1(\sigma')) = \exists x T^-(\varphi)$ with $x = \pi_1(\sigma')$

Observe that in (38-b) the values of σ_1 and σ_2 are irrelevant: these are alphabetic variants of the two parts of σ. As should be clear from the concept of alphabetic innocence, the variables chosen for the parts of the conjunction are no more relevant once the parts enter the larger expression, in which they only serve to determine the ϵ/ν-structure of its constituents. Therefore it does not hold that for any conjunction α in ARPL, $\alpha = T(T^-(\alpha))$, whereas for any α in PL it holds that $\alpha = T^-(T(\alpha))$.

We will now show that the translation from PL to ARPL preserves truth conditions. In order to do so we first define a function $GtoS$ which converts the representation of truth conditions in PL, namely a set of assignment functions, into a set of finite sequences as used in the semantic representation of ARPL. Recall that due to the coincidence lemma the truth conditions of a formula A in terms of sets of assignment functions g only depend on the values for free variables that occur in A. Moreover, the set of these variables is precisely those that occur in $\pi_2(T(A))$ (proof by induction) which also states the order of occurences of these variables and the arity of $\pi_1(T(A))$.

Let $\vec{x}$ denote a sequence of variables $x_1 \ldots x_n$ and define $g(\vec{x}) := \langle g(x_1), \ldots, g(x_n) \rangle$. We now define for any formula $A \in PL$:

(39) $GtoS(A) := \{s : \text{there is a } g \in [\![A]\!] \text{ such that } s = g(\pi_2(T(A)))\}$

Note that if A is false, there is no such g, hence the condition on s cannot be satisfied and $GtoS(A) = \emptyset$. If A is a true closed sentence, there is such a g but $x_1 \ldots x_n = \Lambda$, hence $GtoS(A) = \{\Lambda\} = \{\emptyset\}$. Furthermore, if A is true, then the restriction of s to $\sigma = \pi_2(T(A))$ becomes irrelevant, hence $GtoS(A) = D^n$.

In order to prove the equivalence of PL and ARPL we state the following:

(40) **Lemma 1:** $GtoS(A) = \pi_1([\![T(A)]\!]) \cap \pi_2([\![T(A)]\!])$

The proof of (40) is stated in Appendix 2. We can now prove the following theorems:

(41) **Theorem 1:** A is satisfiable iff $T(A)$ is satisfiable.

Proof A is satisfiable iff $[\![A]\!] \neq \emptyset$ iff $GtoS(A) \neq \emptyset$ iff $\pi_1([\![T(A)]\!]) \cap \pi_2([\![T(A)]\!]) \neq \emptyset$ iff $T(A)$ is satisfiable. $\square$

(42) **Theorem 2:** A is true iff $T(A)$ is true. A is true iff $\neg A$ is not satisfiable iff (by Theorem 1) $T(\neg A)$ is not satisfiable iff $\pi_1([\![\neg A]\!]) \cap \pi_2([\![\neg A]\!]) = \emptyset$ iff $\pi_1([\![\neg A]\!]) \cap \pi_2([\![A]\!]) = \emptyset$ iff $D^n \setminus \pi_1([\![A]\!]) \cap \pi_2([\![A]\!]) = \emptyset$ iff $\pi_2([\![A]\!]) \subseteq \pi_1([\![A]\!])$ iff $T(A)$ is true. $\square$

Due to these equivalences, the classical notions for PL as given in (43) still hold for ARPL:

(43) a. $A \models B$ iff for any interpretation I, if A is true in I, then B is true.
b. For any set Σ of formulas, $\Sigma \models B$ iff for any interpretation I, if every $A \in \Sigma$ is true, then B is true.
c. A formula is valid iff it is true in all interpretations.

As for modus ponens, PL-theories differ depending on whether or not open formulas can be elements of theories. If so, open formulas are semantically equivalent to universally quantified formulas, so that

(44) $\varphi \vdash \forall x \varphi$

is a valid inference rule. And the same holds when translating the formulas into ARPL or APPL, assuming again that $\forall$ is defined in terms of negation and existential quantification. On the other hand, the Deduction Theorem cannot hold in full generality, since (44) should not imply the derivability (and logical validity) of $\vdash \varphi \rightarrow \forall x \varphi$, unless quantification is vacuous. But this is exactly parallel to classical PL as detailed in e.g. [10]:

(45) If a deduction $\Gamma, A \vdash B$ involves no application of (44) of which the quantified variable is free in A, then $\Gamma \vdash A \rightarrow B$.

See [10] p. 60ff for details.

As a consequence of alphabetic innocence, it now holds that indeed $A \models B$ when A and B are alphabetic variants; we thus need a new inference rule which is not valid in classical PL:

(46) $A \vdash B$ if A and B are alphabetic variants.

However, the Deductions Theorem must be blocked for (46) as well; hence (45) must be extended to:

(47) If a deduction $\Gamma, A \vdash B$ involves no application of (46) or (44) of which the quantified variable is free in A, then $\Gamma \vdash A \rightarrow B$

For example, $\langle\langle P, x\rangle \rightarrow \langle P, y\rangle, xy\rangle$ should not be derivable from $\langle P, x\rangle \vdash \langle P, y\rangle$, because this formula is not a tautology: assume $D = \{a, b\}$ and $I(P) = \{a\}$. Then $[\![\langle \neg P, x\rangle]\!] = \langle \{b\}, D\rangle$, $[\![\langle(\langle P, x\rangle \wedge \langle \neg P, y\rangle), xy\rangle]\!] = \langle\{\langle a, b\rangle\}, D^2\rangle$ and

$[\![\neg\langle\langle P, x\rangle \wedge \langle\neg P, y\rangle, xy\rangle]\!] = [\![\langle\langle P, x\rangle \rightarrow \langle P, y\rangle, xy\rangle]\!] = \langle\{\langle b, a\rangle\}, D^2\rangle$. This formula is not even true in the chosen model.

4.2 A Note on Equivalence of Open Formulas

As the reader might have noticed the usual notions in (43) are defined in terms of truth rather than satisfiability. This is not essential, as one could also equivalently define these notions based in satisfiability. However, satisfiability in classical PL gives rise to a more fine grained notion of equivalence: A and B are extensionally equivalent in a model M iff $[\![A]\!]_M = [\![B]\!]_M$. For examply, in every model $(P(x) \wedge Q(y))$ and $(Q(y) \wedge P(x))$ are extensionally equivalent, as they denote the same sets of assignments. Likewise, one might want to say that $T((P(x) \wedge Q(y)))$ and $T((Q(y) \wedge P(x)))$ are extensionally equivalent in a model, but it is not obvious how this could be done, as the two formulas denote different objects in ARPL.[7]

In order to express local equivalence in the present framework it seems necesary to appeal to intensional concepts by exploiting the fact that the interpretation function I *uniquely* determines both $T(A)$ and $T(B)$ via the constants that appear in A and B. Hence, any interpretation J that yields the same extension as I for $T(A)$ (and thus differs from I in inessential ways, eg. for constants that do not appear in A) will also yield the same extension for $T(B)$. In other words, the interpretations that coincide with I on the first formula are exactly the ones that do so on the second.

In PL, we may say that *A locally implies B* in a model $M = \langle D, I\rangle$ iff $[\![A]\!]_I \subseteq [\![B]\!]_I$. We propose to define the corresponding notion in ARPL by the following definition:

(48) For any I, B is a local implication of A in I iff
$\{J : [\![A]\!]_J$ is satisfiable and $[\![A]\!]_J = [\![A]\!]_I\} \subseteq$
$\{K : [\![B]\!]_K$ is satisfiable and $[\![B]\!]_K = [\![B]\!]_I\}$

Of course, if $[\![A]\!]_J$ is not satisfiable then A is false in J (and I); this case is trivial, as everything follows from falsity. Moreover, it is trivial that $(P(x) \wedge Q(y))$ is a logical consequence of $(Q(y) \wedge P(x))$ and vice versa, because in PL it holds that $[\![(P(x) \wedge Q(y))]\!]_I = [\![A]\!]_I = [\![B]\!]_I = [\![(Q(y) \wedge P(x))]\!]_I$. This is different for $T(A)$ and $T(B)$ in ARPL/APPL. Here, $T(A) = \langle (P(x) \wedge Q(y)), xy\rangle$ expresses a certain relation based on the interpretation I of the constants of PL. If the formulas were not equivalent we have to find an interpretation function that gives the same extension to $T(A)$ as I does, but an extension to $T(B)$ that differs from that I gives to $T(B)$. Clearly this is impossible, hence the local equivalence can be established without

[7]Note that the problem does not affect global logical relations: it still holds that $T((P(x) \wedge Q(y))) \vdash T((Q(y) \wedge P(x)))$ and $T((P(x) \wedge Q(y))) \models T((Q(y) \wedge P(x)))$ because we quantify over all possible models.

reference to assignment sets. We leave it to the reader to show that A locally implies B in I (in PL) iff B is a local implication of A in I (in ARPL).

5 Arity Preserving Predicate Logic (APPL)

Admittedly, arity reduction is a cumbersome process and one might wonder whether such a complication is really necessary. As it turns out, arity reduction can be avoided but at a price. In this section we turn to the consequences of such a system, showing which compensatory complications arise. The impatient reader may skip this section.

The idea of keeping the arity of a formula untouched by quantification is a natural one once we suppose that a formula of arity n is true iff it denotes D^n and false iff it denotes the empty set. It seems, then, that the only change we have to make concerns quantification. Suppose, existential quantification were not arity reducing. This leads to the following straightforward definition [8]:

(49)
$$\begin{aligned} [\![f_\exists(\langle\varphi, k\sigma'\rangle)]\!] &= \mathcal{O}_\exists([\![\varphi]\!], [\![k\sigma']\!]) \\ &= \langle\{s \in D^n,\ n \text{ the arity of } \pi_1([\![\varphi]\!]) : \text{ there is an } s' \text{ such that} \\ &\quad s' \in \pi_1([\![\varphi]\!]), \\ &\quad \text{if } \langle\langle 1, i\rangle, \epsilon\rangle \in [\![k\sigma]\!] \text{ and } \langle\langle 1, j\rangle, \epsilon\rangle \in [\![k\sigma]\!] \text{ then } \pi_i(s') = \pi_j(s'), \\ &\quad \text{and for all } i \le n,\ \pi_i(s) = \pi_i(s') \text{ unless} \\ &\quad \langle\langle 1, i+1\rangle, \epsilon\rangle \in [\![k\sigma']\!])\}, [\![\sigma']\!]_D\rangle \end{aligned}$$

As usual, the definition says that s can take any value at a position i that is bound by k but otherwise must be identifical to some s' that satisfies the relational part of φ. If there is no such s, the set is empty (the formula is false). As expected, a formula $\varphi \in Fml_n$ is true iff $\pi_1([\![\varphi]\!]) = D^n$.

However, (49) has some unusual side-effects. Assume A and B are closed formulas, say of arity 1. According to the scheme for conjunction, $(A \wedge B)$ has to be supplied with a coordination scheme. This could be $\bigcirc\bigcirc$ or $\bigcirc\square$. Depending on this choice we get different denotations. This is somewhat disturbing although the difference is simply immaterial for the truth conditions, as the reader may easily verify.

What is more disturbing is the fact that we now have no easy way to state a translation T^- into PL. The problem is that we do not see from just inspecting the coordination whether the variables represented there are free or bound. Bound variables are irrelevant, but free variables cannot simply be ignored. Of course we can find out by starting a complicated recursive research into the structures in A and B. But this is an additional complication that replicates the complication induced by arity reduction.

Of course, if we could somehow mark variables as bound during the process of composing formulas, this difficulty can be avoided because we can apply the same

[8]Note that this condition can be dispensed with: if it should be the case that s has different objects in positions with the same variable x bound by the coordination, then such an s will not be an element of $[\![\sigma']\!]_D$ and hence will be irrelevant. In the revised version below, however, the condition will be relevant, as $[\![\sigma']\!]_D$ will not contain information about coordinated bound variables.

translation function as before but only have to ignore bound variables. This is in fact what we propose in this section: we will attain our target by using different symbols for free and bound variables. The different variables are said to have a different sort or "color" so that a variable will change its "color" as soon as it is bound. As a side effect, the difference between the colors will be employed when saying that bound variables simply cannot be coordinated, hence the above coordination $\bigcirc\bigcirc$ will not be well-formed unless $\bigcirc$ is a free variable. Hence, only free variables can be coordinated.

One way of implementing "colors" is to use K and add the set $\{+, -\}$ as follows:

(50) **Definition of K-symbol:** (version 1)
Let $K = \{\bigcirc, \Box, \Diamond, \ldots\}$. A K-symbol is an element of $\{\langle x, y\rangle : x \in \{\bigcirc, \Box, \Diamond, \ldots\}, y \in \{+, -\}\}$.

An alternative is to introduce two disjoint vocabularies K_F (e.g. hollow symbols) and K_B (black symbols) for free and bound variables respectively:

(51) **Definition of K-symbol:** (version 2)
Let K_F and K_B be denumerably infinite disjoint set of symbols (e.g. $\{\bigcirc, \Box, \Diamond, \triangledown, \vartriangle, \ldots\}$ and $\{$●, ■,◆,▼,▲,...$\}$). A K-symbol is an element of $\mathrm{K}_F \cup \mathrm{K}_B$.

As the two formulations of the distinction are notational variants nothing hinges on the choice; the second variant will be used in the following as it leads to a straightforward modification of the definition of a K-sequence.

We now have to make sure that:

a. the newly built formulas are interpreted with respect to appropriate (modified) ϵ/ν-structures that reflect the difference between K_F and K_B in semantics;
b. binding by a quantifier introduces a new (black) symbol that cannot be coordinated by any further operation.

Let us first turn to the redefinition of ϵ/ν-structures. In order to make a difference between two types of variables, we may exploit a redundancy in the previous definition of ϵ/ν-structures. Note that due to (8-a) it is impossible that $\langle\langle i, i\rangle, \nu\rangle$. But now assume that we modify the definition of ϵ/ν-structures in such a way that this case is now permitted. The convention is that $\langle\langle i, i\rangle, \nu\rangle$ is the "denotation" of a bound variable and $\langle\langle i, i\rangle, \epsilon\rangle$ that of a free variable. More explicitly, we now require that

(52) **Definition of ϵ/ν-structure, revised:**
κ is an ϵ/ν-structure of rank n iff κ is a function from all pairs of integers i, j with $1 \leq j \leq i \leq n$ into the set $\{\epsilon, \nu\}$ such that for all i, j, k:

a. Coordination of positions implies identity of color:
if $\kappa(i, j) = \epsilon$, then $\kappa(i, i) = \kappa(j, j)$,
b. Transitivity:
if $\kappa(i, j) = \epsilon$ and $\kappa(j, k) = \epsilon$, then $\kappa(i, k) = \epsilon$.
c. Well-formedness condition: If $\kappa(i, i) = \nu$, then $\kappa(i, j) = \nu$ for all j.

Let us next provide for the modified syntactic notions. The guiding intuition is that the type of variable cannot be changed when considering t-equivalent sequences, unless there is a process called binding that turns a free variable into a bound one. First assume that we define t-equivalence as in (13). Then ambiguous merge cannot change the coloring of its constituents. The only possible and in fact obligatory change is by binding a free variable. This is largely a matter of syntax. For example, if $\varphi = \langle P, \bigcirc\square\blacklozenge\triangledown\rangle$, then $\langle\exists\varphi, \bigcirc\square\blacklozenge\blacktriangledown\rangle$ is well-formed and we can tell from the difference between $\bigcirc\square\blacklozenge\triangledown$ and $\bigcirc\square\blacklozenge\blacktriangledown$ that the last position has been bound. We thus define the syntax of existential quantification as follows:

(53) If $\varphi = \langle\psi, \sigma\rangle \in Fml_n$, then

$$f_\exists(\varphi, \sigma') = \langle\exists\varphi, \sigma'\rangle$$

for any σ' such that:
there is a $k \in K^1_F$ and a disambiguation $s \in k@\sigma$ such that, if k targets a position i in s and $\pi_i(s) \in K^1_F$, then $\pi_{i+1}(\sigma') \in K^1_B$; otherwise $\pi_j(\sigma') = \pi_j(\sigma)$ for all other $j \le n$.

It now follows that binding always creates variables that can never be coordinated; hence the situation described at the beginning of this paragraph can never arise. We only have to adjust the truth conditions as follows:

(54)
$$\begin{aligned}[\![f_\exists(\langle\psi, \sigma\rangle, \sigma'\rangle)]\!] &= \mathcal{O}_\exists([\![\langle\psi, \sigma\rangle]\!], [\![\sigma']\!])\\ &= \langle\{s \in D^n : \text{there is an } s' \in \pi_1([\![\langle\psi, \sigma\rangle]\!]) \text{ such that}\\ &\quad \text{if } \langle\langle i, j\rangle, \epsilon\rangle \in [\![\sigma]\!],\ \langle\langle i, i\rangle, \epsilon\rangle \in [\![\sigma]\!] \text{ and } \langle\langle i, i\rangle, \nu\rangle \in [\![\sigma']\!]\\ &\qquad \text{then } \pi_i(s') = \pi_j(s'),\\ &\quad \text{and for all } i \le n, \pi_i(s) = \pi_i(s')\\ &\quad \text{unless } \langle\langle i, i\rangle, \nu\rangle \in [\![\sigma]\!]\}, [\![\sigma']\!]_D\rangle\end{aligned}$$

Intuitively, this means that we first assemble all s which coincide with some s' on the value of free variables, where s' satisfies the relation φ, the values for bound variables can be arbitrary, as is the case for the corresponding assignment functions. As this might still disregard free variables that are coordinated, we add $[\![\sigma']\!]_D$ as the second component of the denotation.

6 Adding Constants and Functions

As constants and functions occur intertwined with variables in argument sequences we will have to revise a number of definitions. First we will define recursively the sequences whose counterpart in PL is the sequence of arguments that contain constants or functions, as in $P(x_1, \dots\ c, \dots f(y_1, \dots y_m), \dots\ x_n)$. As before, the variables in $f(y_1, \dots\ y_m)$ must come along with a desambiguation that coordinates them with other variables in the same formula.

Second, we will have to add to our semantics the interpetation of such sequences. Note that previously, the interpretation was ultimately a sequence s of individuals in the model that conforms to the ϵ/ν-structure of the sequence of variables. We now have to account for the fact that this sequence can be "impure" by containing other expressions than just variables.

In the syntax we will define a sequence of arguments recursively, each being accompanied by a K-sequence that functions as a desambiguation. A sequence of arguments of length n, alongside with a K-sequence, is defined as AS$_n$ by the following recursion:

(55) Syntax of Arguments:

a. If $k \in K$, then $\langle k, k\rangle \in \text{AS}_1$.
b. if c is a constant, then $\langle c, \Lambda\rangle \in \text{AS}_1$.
c. If $\langle \alpha, \sigma\rangle \in \text{AS}_n$ and $\langle \alpha', \sigma'\rangle$ is an AS$_1$ as defined in (a.) or (b.), then $\langle \alpha + \alpha', \sigma\sigma'\rangle \in \text{AS}_{n+1}$.
d. If $\langle \alpha, \sigma\rangle \in \text{AS}_n$ and f is an n-place function symbol, then $\langle f(\langle \alpha, \sigma\rangle), \sigma\rangle \in \text{AS}_1$.
e. If $\langle \alpha, \sigma_1\rangle \in \text{AS}_n$ and $\langle \alpha', \sigma_2\rangle$ is an AS$_1$ as defined in (d.), then $\langle \alpha + \alpha', \sigma\rangle \in \text{AS}_{n+1}$, where $\sigma \in \sigma_1 @ \sigma_2$.
f. If $\langle \alpha, \sigma\rangle \in \text{AS}_n$ and P is an n-place predicate, then $P\langle \alpha, \sigma\rangle$ is a formula.

Observe that if $\langle \alpha, \sigma\rangle \in \text{AS}_n$, then σ is a K-sequence that reflects the order and number of free variables in α.

Let us now turn to the semantics. We define the denotation or extension of elements in AS$_n$ with respect to a sequence of individuals of the model; this sequence formally plays the role of an assignment function in classical logic. Note however, that it gives directly values for variables, but the variables themselves are not mentioned in the definition, only their ϵ/ν-structures are.

Corresponding to (55-f), we have the following definition:

(56) If $P\langle \alpha, \sigma\rangle$ is a formula, then $[\![P\langle \alpha, \sigma\rangle]\!] = \langle \{s : [\![\alpha]\!]^s \in I(P)\}, [\![\sigma]\!]_D\rangle$.

In the subsequent definition of $[\![\alpha]\!]^s$, the length of s will not match with the number of terms in $\langle t_1, \ldots\ t_n\rangle$, rather it corresponds to the number of free variables in $\langle t_1, \ldots\ t_n\rangle$. This number is determined by the K-sequence σ by definition. We denote by $s - 1$ the sequence like s without the last element of s, $s - n$ the sequence like s without the last n elements, and $s \mid n$ the sequence of the last n elements of s. It holds that $s = (s - n) + s \mid n$. We can now define $[\![\alpha]\!]^s$ by recursion on the length of α:

(57) $[\![\langle t_1, \ldots\ t_n\rangle]\!]^s =$

a. $[\![\langle\langle t_1, \ldots\ t_{n-1}\rangle]\!]^{s-1} + s \mid 1$ if $t_n \in K$.
b. $[\![\langle\langle t_1, \ldots\ t_{n-1}\rangle]\!]^{s} + I(t_n)$ if t_n is a constant.
c. $[\![\langle\langle t_1, \ldots\ t_{n-1}\rangle]\!]^{s-n} + I(f)([\![\alpha]\!]^{s|i})$ if $t_n = f(\langle \alpha, \sigma\rangle))$ and i is the length of σ.

(57-a) shows that s effectively supplies a value for a variable; (57-b) says that s ignores constants, and (57-c) says that the last i elements of s supply the values for the free variables that occur in the scope of the function f. Of course all this is recursive, so that functions can be embedded into functions.

If follows that an expression like $P(c, x, d)$ denotes a 1-place predicate (given that c and d are constants), whereas $P(x, f(y, z), w)$ denotes a 4-place relation. This requires some minor adjustments in previous definitions that involve the arity of formulas and predicates, but nothing essential is at stake and we leave it to the reader to make the required changes.

7 Linguistic Semantics

In this section we will briefly sketch a solution to a conceptual problem in Montague Grammar that will no more arise in the framework of alphabetic innocence. The problem shows up with one of the most elementary operations of grammar, namely what is called "Predicate Modification" (PM) in the Heim/Kratzer textbook [3]. Intuitively, this is a semantic rule that combines the denotation of a noun (or noun phrase) with that of an adjective (or adjective phrase) by intersection of properties; the rule can thus informally be stated as

(58) $\lambda x(\mathrm{AP}(x) \wedge \mathrm{NP}(x))$.

The same also works for the combination of a NP with a relative clause. However, as pointed out in Thomason's footnote 12 on page 261 of [12], more complex phrases could contain free variables, so that the variable x introduced by "PM" must be chosen in such a way as to avoid "accidental binding". As observed by Thomason, this might require renaming of variables that were chosen as the translation for the pronoun. For example, the rule as stated above cannot apply to an AP of the form $\lambda y A(y, x)$, because the result would be $\lambda x(AP(x, x) \wedge NP(x))$ instead of the intended $\lambda x(AP(x, z) \wedge NP(x))$. This is a severe problem, as it makes translations from Natural Language into Intensional Logic sensitive to alphabetic variants and moreover challenges compositionality.

Now, in order to make a comparison between Montagues system and ours it is temping to catch intensionality by a 2-typed version of Montague Grammar and then add lambda expressions to an alphabetically innocent formal language that includes λ-terms along the lines sketched above in Section 6. However, we will argue that such a step, although possible, is not particularly useful, contrary to what Montague Grammar seems to suggest.

There are a number of a priori reasons that tell against including lambda terms into the analysis. First note that the function of lambda abstraction when added to PL is to form properties from open propositions; such a step is of course unnecessary when open propositions already denote properties, as in ARPL and APPL. Second, as already noted by Fine on p. 21, adding expressions like $\lambda x_1 \ldots x_n P(y_1 \ldots y_m)$ would of course necessitate coordination of variables, and the same would hold for functional application. As we will see below, this step is completely redundant, as some sort of identification of positions is already required with each syntactic merge operation.

Let us now change perspective by switching to a lambda free framework. The most elementary question is how verbs combine with arguments. Let us assume that the denotation of an n-place verb is given by an open proposition, i.e. an expression

$\langle P, \sigma\rangle$ which denotes an n-place relation. Adding an argument reduces the arity of that expression, so we may assume a version of ARPL. The argument epression can either be a quantifying expression or a name, i.e. a constant of the language of PL. For the sake of simplicity, let us only consider constants. (In a much too long version of this paper we also developed a system with quantifiers equivalent to [11].) We now need a rule that combines $[\![\langle P, \sigma\rangle]\!]$ with $I(c)$. The rule can be stated similar to quantification, as both rules target a certain position that is determined by the desambiguation σ' and the first position of σ':

(59) $$\mathcal{O}_{con}([\![c]\!], [\![\varphi]\!], [\![\sigma']\!]) = \langle\{t : \text{ there is an } s \in D \oplus \pi_1([\![\varphi]\!]) \text{ such that } t = r_{[\![\sigma]\!]}(s) \text{ and } \forall i([\![\sigma']\!](1, i) = \epsilon \rightarrow \pi_i(s) = I(c)\}, \{t' : \text{ there is an } s' \in [\![\sigma]\!]_D : t' = r_{[\![\sigma]\!]}(s'))\}\rangle$$

where σ' is a desambiguation in $k@\sigma$. But which one? At this point it is important to realize that the order of arguments in $P(x_1 \ldots x_n)$ is determined by a linguistic convention, namely that the first argument corresponds to the highest argument (the subject) of the sentence, the second corresponds to the second etc. This is different with "Curried" relations where P would have to apply to x_n first. Note that these considerations are also relevant when using lambda prefixes; in all cases we follow conventions because, as already observed above, the (syntactic) order of lambdas (and that of arguments) is not strictly speaking part of the meaning of the verb; cf. [7] and [14] p. 80ff. But once an order is given, it is clear that adding a constant corresponds to functional application of a Curried predicate, hence applies to x_n as the first argument, which is the last argument in a traditional representation like $P(x, y, z)$. In other words, the rule for adding constants in ARPL has to target the *last* position in that order, i.e. the last free variable to which the rule can have access. Accordingly, σ' is $zxyz$ and in general σ' is uniquely determined by σ, up to alphabetic variance.

It thus follows that $\mathcal{O}_{con}([\![c]\!], [\![\langle P, xyz\rangle]\!], [\![\sigma']\!]) = [\![\langle\langle P, xyc\rangle, xy\rangle]\!]$ in a language with constants and arity reduction.

Turning back to the rule of predicate modification, note that in its by now classical version the rule involves two aspects determined by the format of the rule. First (58) identifies two argument position by appying P and Q to the *same* variable x. Second it says *which* arguments of P and Q are identified; this simply follows from the assumption that P and Q are of type $\langle et\rangle$. These two assumptions clearly translate into the present framework by first saying that *some* argument positions of the predicates $\langle P, \sigma\rangle$ and $\langle Q, \sigma'\rangle$ must be identified by a desambiguation, and second by saying *which* arguments are involved: these arguments are the *first* positions of σ and σ'.

The first position does not necessarily coincide with the last position of the predicates involved in "PM", because the translation of pronouns as free variables will not change the arity of a relation. For example, *man such that he loves her* will be translated as a two place relation in the present framework, rather than into a one place relation (or its characteristic function) with a free varible for *her*, as in Montague Grammar. On the other hand, we have to keep track of syntactic argument positions, to the effect that the next argument that combines with *love her* is the first argument of the relation, rather than the second. Hence the second argument must again be "colored" in order to make it inaccessable for argument saturation.

The problem of a variable dependent semantics for "PM" has now been solved in a natural way. The solution lies in the fact that the coordination of conjunction cannot change the variables of its conjuncts, and hence cannot produce arbitrary argument coordinations, as opposed to expressions like $\lambda x([\lambda y.P(y,x)](x) \wedge [\lambda y Q(y)](x))$ that end up with "accidentally" identifying P's argument positions. If we want argument identification within a single predicate, we either have to chose identical variables right from the scratch, or we can identify argument position by special operators that do the job of introducing appropriate K-sequences. Other possibilities, e.g. the use of identity relations in the object language, exist. But coordination crucially leaves the structure of its coordinates intact. Hence "arbitrary binding" is ruled out.

Of course many other issues arise, but in the interest of conciseness we only make one point that already suggests itself by generalizing the above examples: In each case, a syntactic merge operation corresponds to a semantic one that requires the identification of argument positions, over and above the usual identification of arguments needed to solve pronominal anaphora. By the linguistic conventions adopted already in classical PL, the identification of these positions in argument saturation is rule governed and always determined as an operation at the left edge of a working space that is determined by syntax. Moreover, all operations are compositional, as the required operations on variable sequences can be formulated in such a way that they always refer to the first or last element of such a sequence of variables. We never have to access an intermediate position and hence do not need arbitrary many semantic operations, much in the spirit of [13].

8 Final Remarks

Concerning the last reference, we want to stress that we are fully aware of alternative variable free semantics (cf. [5], p. 6). In a rejection of a previous version of this article, we were urged to compare our system to a variable free one, where pronouns are not translated as free variables. For example, a reviewer asks, "Why does Fine himself want variables?" and then complaints that Fine and us fail to consider combinatory logic. Thus, our system "inherits sound and insightful reasoning from Fine regarding what one should do if one lives in a world with both free and bound variables, but not a particularly good or deep motivation for why one wants to live in that world".

We insists that a comparison to a variable free systems is besides the point; the systems are orthogonal to each other. Implicit in the reviewers' request is the requirement to demonstrate that the present system be superior to a variable free one. But this is impossible to prove on merely empirical grounds, while conceptual issues might be largely a matter of taste.

Finally, a different rejecting reviewer wanted us to compare our system to de Bruijn' indices, implying that the problems we were attacking are already solved in that system. Again, we deny that this is an option; for the sake of explicitness, we confine the following Appendix to a brief discussion of the issue.

Appendix 1: de Bruijn's indices

DeBruijn indices [1] are a way of representing (primarily) bound variables by using integer indices instead of variable names. The basic idea is that the index $[n]$ refers to a variable bound by the n-th binding expression counting outward from the occurrence of $[n]$. For example, to represent the expression $\lambda x_3.\lambda x_5.x_5$ by means of de Bruijn indices, we first remove the occurrences of x_3 and x_5 immediately following λ (getting $\lambda.\lambda.x_5$), and then we replace x_5 by the corresponding de Bruijn index. Since (the last occurrence of) x_5 in $\lambda x_3.\lambda x_5.x_5$ is bound by λx_5, which is the first λ binder having scope over x_5, we replace x_5 by the deBruijn index [1], resulting in $\lambda.\lambda.[1]$. On the other hand, the term $\lambda x_3.\lambda x_5.x_3$ is represented by $\lambda.\lambda.[2]$, because x_3 is bound by λx_3, which is the second λ-binder having scope over x_3. Importantly, note that the terms $\lambda x_3.\lambda x_5.x_5$ and $\lambda x_3.\lambda x_7.x_7$ are represented by the same term $\lambda.\lambda.[1]$.

DeBruijn indices can also be used to represent free variables. For example, the term $\lambda x_3.x_1$ can be represented by the term $\lambda.[2]$. So, by convention, a deBruijn index $[n]$ stands for a free variable if it is bigger than the number of binders in whose scope it is. In addition, one can stipulate that if a deBrujn index $[n]$ is bigger than the number b of binders having scope over it, then $[n]$ stands for the free variable x_{n-b}. So in the term $\lambda.\lambda.[4]$ the number of binders is 2, the deBrujn index is bigger than 2, and therefore it stands for the free variable x_{4-2}, ie. x_2, whereas in $\lambda.\lambda.[3]$ the deBrujn index stands for the free variable x_1.

Note, importantly, that the terms $\lambda.\lambda.[4]$ and $\lambda.\lambda.[3]$ are not alphabetically innocent: prefixing another λ binder to $\lambda.\lambda.[4]$ results in $\lambda.\lambda.\lambda.[4]$ where, according to our convention [4] stands for the free variable x_1, whereas prefixing $\lambda.\lambda.[3]$ results in $\lambda.\lambda.\lambda.[3]$ where [3] is bound by outermost λ binder, showing that $\lambda.\lambda.[4]$ and $\lambda.\lambda.[3]$ are not alphabetically innocent.

We therefore conclude that deBruijn indices do not provide a solution to the problem addressed in this paper, namely how to define an alphabetically innocent and compositional predicate logic.

Appendix 2: Proof of Lemma 1

Let $T_1(A) = \pi_1(T(A))$ and $T_2(A) = \pi_2(T(A))$. Let $[\![T_1(A)]\!] = \pi_1([\![T_1(A)]\!])$ and $[\![T_2(A)]\!] = \pi_2([\![T_2(A)]\!])$. Recall:

(60) $GtoS(A) := \{s : \text{there is a } g \in [\![A]\!] \text{ and } s = g(T_2(A))\}$

(61) **Lemma 1:** $GtoS(A) = \pi_1([\![T(A)]\!]) \cap \pi_2([\![T(A)]\!])$

Proof of Lemma one by induction over the complexity of formulas.

Atomic Formulas

Let A be an atomic formua $P(x_1, \ldots x_n)$. Then

(62) $GtoS(P(x_1, \ldots x_n))$
$= \{s : \text{there is a } g \in [\![P(x_1, \ldots x_n)]\!] \text{ such that } s = g(T_2(P((x_1, \ldots x_n))))\}$
$= \{s : \text{there is a } g \in [\![P(x_1, \ldots x_n)]\!] \text{ such that } s = \langle g(x_1), \ldots, g(x_n)\rangle\}$

$= \{s : \text{there is a } g, \langle g(x_1), \ldots, g(x_n)\rangle \in I(P) \text{ and } s = \langle g(x_1), \ldots, g(x_n)\rangle\}$
$= \{s : s \in I(P) \text{ and } s \in [\![x_1 \ldots x_n]\!]_D\}$
$= I(P) \cap [\![x_1 \ldots, x_n]\!]_D$
$= \pi_1([\![T(P(x_1, \ldots, x_n))]\!]) \cap \pi_2([\![T(P(x_1, \ldots, x_n))]\!])$

Negation

Recall that $T_2(\neg A) = T_2(A)$ and that $\pi_1[\![\neg A]\!] = D \backslash \pi_1[\![\neg A]\!]$.

(63) $s \in (\pi_1([\![T(\neg A)]\!]) \cap \pi_2([\![T(\neg A)]\!])$

a. iff
$s \in D^n \backslash [\![T_1(A)]\!] \cap [\![T_2(A)]\!]$
b. iff
$s \in D^n \backslash ([\![T_1(A)]\!] \cap [\![T_2(A)]\!]) \cap [\![T_2(\neg A)]\!]$
c. iff
$s \in D^n \backslash GtoS(A) \cap [\![T_2(\neg A)]\!]$
d. iff
$s \in D^n \backslash \{s'$: there is a $g \in [\![A]\!]$ such that$s' = g(T_2(A))$ and s conforms to the free variables of $\neg A$
e. iff
there is no g, $g \in [\![A]\!]$ such that$s = g(T_2(A))\}$ and s conforms to the free variables of $\neg A$
f. iff
for all g, if $s = g(T_2(A))$ then $g \notin [\![A]\!]$, and s conforms to the free variables of $\neg A$
g. iff
there is a $g \in [\![\neg A]\!]$ such that $s = g(T_2(\neg A))$
h. iff
$s \in GtoS(\neg A)$

Comments: In (63-b) we duplicate the condition that s conforms to the free variables; as can be seen at the end of the derivation the second condition becomes redundant and is elimated in the step from f. to g.

Conjunction

(64) $s \in GtoS(A \wedge B)$

a. iff
there is a g, $g \in [\![A \wedge B]\!]$ and $s = g(T_2(A \wedge B))$
b. iff
$s = s_1 s_2$ and there is a g, $g \in [\![A]\!]$, $g \in [\![B]\!]$, and $s_1 s_2 = g(T_2(A \wedge B)) = g(T_2(A)) + g(T_2(B))\}$
c. iff
there is a g, $g \in [\![A]\!]$, $g \in [\![B]\!]$, and $s_1 = g(T_2(A))$, $s_2 = g(T_2(B))$ and $s_1 s_2 \in [\![T_2(A \wedge B)]\!]_D$

d. iff
there is a h, $h \in [\![A]\!]$, $s_1 = g(T_2(A))$ and there is an f, $f \in [\![B]\!]$, $s_2 = f(T_2(B))$ and $s_1s_2 \in [\![T_2(A \wedge B)]\!]_D$

e. iff
$s_1 \in GtoS(A)$, $s_2 \in GtoS(B)$ and $s_1s_2 \in [\![T_2(A \wedge B)]\!]_D$

f. iff
$s_1 \in [\![T_1(A)]\!] \cap [\![T_2(A)]\!]$, $s_2 \in [\![T_1(B)]\!] \cap [\![T_2(B)]\!]$ and $s_1s_2 \in [\![T_2(A \wedge B)]\!]_D$

g. iff
$s_1 \in [\![T_1(A)]\!]$, $s_2 \in [\![T_1(B)]\!]$ and $s_1s_2 \in [\![T_2(A \wedge B)]\!]_D$

h. iff
$s_1s_2 : s_1s_2 \in [\![T_1(A)]\!] \otimes [\![T_1(B)]\!]$ and $s_1s_2 \in [\![T_2(A \wedge B)]\!]_D$

i. iff
$s_1s_2 \in [\![T_1(A \wedge B)]\!]$ and $s_1s_2 \in [\![T_2(A \wedge B)]\!]_D$

j. iff
$s \in [\![T_1(A \wedge B)]\!] \cap [\![T_2(A \wedge B)]\!]_D$

k. iff
$s \in \pi_1([\![T(A \wedge B)]\!]) \cap \pi_2([\![T(A \wedge B)]\!])$

Comments: Most equivalences follow by definition. Ad (64-d) upwards from right to left: assume that h and f assign different values to some variable that occurs both in A and B. Then it would be impossible for s_1s_2 to conform to $T_2(A \wedge B)$. Therefore these values must be identical and we can combine h and f into the g of (64-c).

Quantification

(65) $s \in GtoS(\exists x A)$

a. iff
$s \in [\![T_1(\exists x A)]\!] \cap [\![T_2(\exists x A)]\!]$

b. iff
$s \in [\![T_1(\exists x A)]\!]$ and s conforms to the free variables of $\exists x A$

c. iff
there is an $s' \in D \otimes [\![T_1(A)]\!]$ such that $s = r_{[\![\sigma]\!]}(s')$, s' conforms to x, and s conforms to the free variables of $\exists x A$

d. iff
there is an $s' \in D \otimes [\![T_1(A)]\!]$ such that $s = r_{[\![\sigma]\!]}(s')$, and s' conforms to the free variables of A

e. iff
there is an s'' and an $a \in D$ such that $s'' \in [\![T_1(A)]\!]$, $s = r_{[\![\sigma]\!]}(as'')$, and s'' conforms to the free variables of A

f. iff
there is an s'' and an a such that $s'' \in [\![T_1(A)]\!] \cap [\![T_2(A)]\!]$, $s = r_{[\![\sigma]\!]}(as'')$

g. iff
there is an s'' and an a such that $s'' \in GtoS(A)$, $s = r_{[\![\sigma]\!]}(as'')$

h. iff
   there is an s'', an a, and a g such that $g \in [\![A]\!]$, $s'' = g(T_2(A))$, and $s = r_{[\![\sigma]\!]}(as'')$
i. iff
   there is an a and a g such that $g \in [\![A]\!]$, and $s = r_{[\![\sigma]\!]}(a + g(T_2(A)))$
j. iff
   there is an a, a g and a g' such that $g' \in [\![\exists x A]\!]$, g and g' possibly differ only for values for x, and $s = r_{[\![\sigma]\!]}(a + g(T_2(A)))$.
k. iff
   for some g', $g' \in [\![\exists x A]\!]$, $s = g'(T_2(\exists x A))$
l. iff
$s \in GtoS(\exists x A)$

Comments on (65-k): $g'(T_2(\exists x A)) = r_{[\![\sigma]\!]}(a + g(T_2(A)))$, because the reduction of $g(T_2(A))$ yields exactly the values for the free variables of $T_2(\exists x A)$; moreover, it ignores only the values for x of g, which is exactly what g' does.

References

1. de Bruijn, N.G. (1972). Lambda Calculus notation with nameless dummies, a tool for automatic formula manipulation with application to the Church-Rosser theorem. *Indagationes Mathematicae*, *34*, 381–392.
2. Fine, K. (2007). *Semantic relationism*. Oxford: Blackwell.
3. Heim, I., & Kratzer, A. (1998). *Semantics in generative grammar*. Oxford: Blackwell.
4. Henkin, L., & Tarski, A. (1961). Cylindric algebras. In Dilworth, R. (Ed.) *Lattice theory, proceedings symposium in pure mathematics*, (Vol. 2 p. 83113). Providence: American Mathematical Society.
5. Jacobson, P. (1999). Towards a variable free semantics. *Linguistics and Philosophy*, *22*, 117–185.
6. Jacobson, P. (2007). Direct compositionality and variable-free semantics. In Barker, C., & Jacobson, P. (Eds.) *Direct compositionality* (pp. 191–236). Oxford: Oxford University Press.
7. Kracht, M. (2007). The emergence of syntactic structure. *Linguistics and Philosophy*, *30*, 47–95.
8. Kracht, M. (2011). *Lectures on interpreted languages and compositionality*. Berlin: Springer.
9. Mahler, T. (1993). Morphogrammatik. Eine Einführung in die Theorie der logischen Form. www.thinkartlab.com/pkl/tm/MG-Buch.pdf.
10. Mendelson, E. (1963). *Introduction to mathematical logic*. Princeton: Van Nostrand.
11. Montague, R. (1973). The proper treatment of quantification in ordinary english. In Hintikka, J., & Suppes, P. (Eds.) *Approaches to natural language* (pp. 221–242). Dordrecht: Reidel.
12. Montague, R. (1974). Formal philosophy. Selected papers of Richard Montague. In Thomason, R.H. (Ed.) New Haven/London: Yale University Press.
13. Quine, W.V.O. (1960). Variables explained away. *Proceedings of the American Philosophical Association*, *140*, 343–347.
14. Zimmermann, T.E., & Sternefeld, W. (2013). *Introduction to semantics an essential guide to the composition of meaning*. Berlin: De Gruyter Mouton.