Are Logical Languages Compositional?

Marcus Kracht *
Fakultät Linguistik und Literaturwissenschaft
Universität Bielefeld
Postfach 10 01 31
33501 Bielefeld
marcus.kracht@uni-bielefeld.de

October 9, 2013

Abstract

In this paper I argue that in contrast to natural languages, logical languages typically are *not* compositional. This does not mean that the meaning of expressions cannot be determined at all using some well-defined set of rules. It only means that the meaning of an expression cannot be determined without looking at its form. If one is serious about the compositionality of a logic, the only possibility I see is to define it via abstraction from a variable free language.

1 Introduction

A language is called compositional if the meaning of a complex expression is a function of the meanings of the parts from which it is being made and the mode of composition employed in making it (see below for an exact definition). This is more restrictive than just having *any* procedure to build up the meanings, because the meaning must be the *only* input. Natural languages are thought to be compositional, for it is difficult to see how else one can establish the meaning of new

*I wish to thank the audience of Trends in Logic XI, especially Joop Leo, Sergei Odintsov and Graham Priest as well as an anonymous reviewer for useful discussion.

expressions. Similarly for artificial languages. One might think that logical languages are the best examples of compositional languages. After all, they are the prime examples of artificial languages, they are unambiguous and semantically well-defined.

And yet logical languages have so far defied a compositional analysis despite announcements to the contrary. The problem is not that no scheme of interpretation can be given. The problem is that these schemes typically are not compositional. The only semantics that technically works is based on valuations. However, valuations are not compositional since they encode syntactic identity. Indeed, the more one looks into the matter the harder it is to understand how a proper semantics can be set up that is both true to the original understanding and compositional at the same time.

In this paper I take a fresh look at the issue and try to identify what is going wrong. The upshot is that what thwarts our hopes of providing a compositional account are the variables. Thus, languages can only be compositional if they do not contain variables. The picture that arises is one of logic not as a language with a semantics but as the result of abstracting truth patterns from a language.

2 Preliminaries

Let E be a set of **expressions**. For simplicity, $E = A^*$ for some finite set A of letters. Further, let M be another set, the set of **meanings**. $E \times M$ is the set of **signs**. A **language** is an arbitrary set of signs, in other words, a subset of $E \times M$. A *grammar* is a finite set of partial functions over signs. To be more precise, let F be a finite set of function symbols, called **modes**, and $\Omega : F \to \mathbb{N}$ a signature. Then an Ω-**grammar** is a function $\mathcal{I}$ such that for every $f \in F$, $\mathcal{I}(f)$ is an $\Omega(f)$-ary partial function on $E \times M$. The **language generated** by the grammar is the smallest set $L(\mathcal{I})$ such that for every $f \in F$ and every σ_i, $i < \Omega(f)$, if $\sigma_i \in L(\mathcal{I})$ for every $i < \Omega(f)$ then also $\mathcal{I}(f)(\sigma_0, \cdots, \sigma_{\Omega(f)-1}) \in L$, if that sign exists. This means in particular that for every f such that $\Omega(f) = 0$ (such f is called **lexical**) the sign $\mathcal{I}(f)()$ is in $L(\mathcal{I})$ (again if that sign exists, which we assume to be the case at least for 0-ary function symbols). This guarantees that the set $L(\mathcal{I})$ is not empty as long as there are lexical modes. The nonlexical modes generate the set $L(\mathcal{I})$ in the standard inductive way.

> The set $L(\mathcal{I})$ is the smallest subset of $E \times M$ such that for every $f \in F$ and signs σ_i, $i < \Omega(f)$, if for every $i < n$ $\sigma_i \in L(\mathcal{I})$ then also

$\mathcal{I}(f)(\sigma_0, \cdots, \sigma_{\Omega(f)-1}) \in L(\mathcal{I})$ on condition that $\mathcal{I}(f)$ is defined on this input.

In other words, take the clone of partial term functions generated by the $\mathcal{I}(f)$. Then $L(\mathcal{I})$ is the set of all signs which are the value of some constant term function from that clone.

$\mathcal{I}$ is **compositional** if for every $f \in F$ there is a partial function $f^{\mu} : M^{\Omega(f)} \hookrightarrow M$ such that for every m_i ($i < \Omega(f)$): (i) either there are e_i such that $\mathcal{I}(f)$ is defined on the signs $\langle e_i, m_i \rangle$ and for all such e_i there is an $e \in E$ with

$$\mathcal{I}(f)(\langle e_0, m_0 \rangle, \cdots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1} \rangle) = \langle e, f^{\mu}(m_0, \cdots, m_{\Omega(f)-1}) \rangle \tag{1}$$

or (ii) $\mathcal{I}(f)$ is undefined no matter what e_i get chosen, and then $f^{\mu}(m_0, \cdots, m_{\Omega(f)-1})$ is undefined as well.

A grammar is **autonomous** if for every $f \in F$ there is a partial function $f^{\varepsilon} : E^{\Omega(f)} \hookrightarrow E$ such that for every choice of e_i ($i < \Omega(f)$): (i) either there are m_i such that $\mathcal{I}(f)$ is defined on the $\langle e_i, m_i \rangle$ and for all such m_i there is an $m \in M$ with

$$\mathcal{I}(f)(\langle e_0, m_0 \rangle, \cdots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1} \rangle) = \langle f^{\varepsilon}(e_0, \cdots, e_{\Omega(f)-1}), m \rangle \tag{2}$$

or (ii) $\mathcal{I}(f)$ is undefined no matter what m_i get chosen and then $f^{\varepsilon}(e_0, \cdots, e_{\Omega(f)-1})$ is undefined as well.

Grammars for logical languages are generally unambiguous, so the question of compositionality becomes indistinguishable from the question of *independence*. A grammar is **independent** if it is both autonomous and compositional. This means that for every $f \in F$ there exist partial functions $f^{\varepsilon} : E^{\Omega(f)} \hookrightarrow E$ and $f^{\mu} : M^{\Omega(f)} \hookrightarrow M$ with

$$\begin{aligned} \mathcal{I}(f)(\langle e_0, m_0 \rangle, \cdots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1} \rangle) \\ = \langle f^{\varepsilon}(e_0, \cdots, e_{\Omega(f)-1}), f^{\mu}(m_0, \cdots, m_{\Omega(f)-1}) \rangle \end{aligned} \tag{3}$$

with the proviso that the left hand side is defined if and only if the right hand side is defined. Actually, the most preferred understanding of this is that rather than using $\mathcal{I}$ we associate with f the two partial functions f^{ε} and f^{μ}. This is a stronger notion, since the domain of $\mathcal{I}(f)$ is uniquely defined from the domains of f^{ε} and f^{μ}, while the converse does not hold, see [Kracht, 2011]. This leads to the notion of an **independent bigrammar**, which is a pair of interpretations, $\langle \mathcal{I}^{\varepsilon}, \mathcal{I}^{\mu} \rangle$, such that $\mathcal{I}^{\varepsilon}(f)$ is an $\Omega(f)$-ary partial function on E and $\mathcal{I}^{\mu}(f)$ an $\Omega(f)$-ary partial

function on M. We then let f^ε abbreviate $\mathcal{I}^\varepsilon(f)$ and f^μ abbreviate $\mathcal{I}^\mu(f)$. The bigrammar uniquely defines a grammar $\mathcal{I}$ by

$$\begin{aligned}(4)\qquad \mathcal{I}(f)(\langle e_0, m_0\rangle, \cdots, \langle e_{\Omega(f)-1}, m_{\Omega(f)-1}\rangle)\\ := \langle \mathcal{I}^\varepsilon(f)(e_0, \cdots, e_{\Omega(f)-1}), \mathcal{I}^\mu(f)(m_0, \cdots, m_{\Omega(f)-1})\rangle\end{aligned}$$

with the right hand side being defined if and only if the left hand side is. This is the format we shall assume below.

The signature Ω defines a set of Ω-**terms** in the following way.

1. ξ_i, $i \in \mathbb{N}$, is an Ω-term.
2. If t_i, $i < \Omega(f)$, are Ω-terms, then so is $f(t_0, \cdots, t_{\Omega(f)-1})$.

The ξ_i are variables over analysis terms. A term is called **constant** if it is not built with the help of the ξ_i. Let us define the sign of the term inductively as follows.

$$(5)\qquad (f(t_0, \cdots, t_{\Omega(f)-1}))^{\mathcal{I}} := \mathcal{I}(f)(t_0^{\mathcal{I}}, \cdots, t_{\Omega(f)-1}^{\mathcal{I}})$$

Since we assume our grammars to be bigrammars, we can rewrite this as follows. $t^{\mathcal{I}} = \langle t^\varepsilon, t^\mu\rangle$, where

$$\begin{aligned}(6)\qquad (f(t_0, \cdots, t_{\Omega(f)-1}))^\varepsilon &:= \mathcal{I}^\varepsilon(f)(t_0^\varepsilon, \cdots, t_{\Omega(f)-1}^\varepsilon)\\ (f(t_0, \cdots, t_{\Omega(f)-1}))^\mu &:= \mathcal{I}^\mu(f)(t_0^\mu, \cdots, t_{\Omega(f)-1}^\mu)\end{aligned}$$

Notice that due to the partiality of the functions, the values t^ε, t^μ, and so on, do not always exist. A constant term is therefore called **definite** if $t^{\mathcal{I}}$ exists. The language generated by the grammar $\langle \mathcal{I}^\varepsilon, \mathcal{I}^\mu\rangle$ is the set of all $\langle t^\varepsilon, t^\mu\rangle$ for definite, constant terms t.

Call L **unambiguous** if for all $\langle e, m\rangle, \langle e, m'\rangle \in L$, $m = m'$.

Proposition 1. *If L is unambiguous and compositional it is also independent.*

The idea is this. Suppose e in (1) depends both on e_i and m_i. By assumption, L is a partial function from E to M, and so we may eliminate m_i by using $L(e_i)$ instead. Reformulated in this way e depends only on the e_i.

To show the scope and limits of the notion of compositionality let me state and prove the following fact. Define for a language L the projection

$$(7)\qquad p_1[L] := \{e : \text{ there is } m \in M\colon \langle e, m\rangle \in L\}$$

Proposition 2. *Suppose that M is finite and that $L \subseteq A^* \times M$ is a language such that $p_1[L]$ is at most countable. Then L has an independent bigrammar.*

Proof. Clearly, a finite language has an independent grammar (just define a mode for each sign, and no other modes). So assume that L is infinite. $p_1[L]$ is then infinite (since M is finite). By assumption $p_1[L]$ is countably infinite, and so there is a finite set U of partial functions on A^* that together generate $p_1[L]$. For each $u \in U$ and $m \in M$ create a new mode $g^{u,m}$, whose arity is the arity of u. Now put

$$e^{u,m}(e_0, \cdots, e_{n-1}) := \begin{cases} u(e_0, \cdots, e_{n-1}) & \text{if } u(e_0, \cdots, e_{n-1}) \text{ is defined} \\ & \text{and } \langle u(e_0, \cdots, e_{n-1}), m\rangle \in L, \\ \text{undefined} & \text{else.} \end{cases} \tag{8}$$

Also, let $c^{u,m}(m_0, \cdots, m_{n-1}) := m$. Finally, let $\mathcal{I}^{\varepsilon}(g^{u,m}) := e^{u,m}$ and $\mathcal{I}^{\mu}(g^{u,m}) := c^{u,m}$. The bigrammar $\langle \mathcal{I}^{\varepsilon}, \mathcal{I}^{\mu}\rangle$ is obviously independent, and it is straightforward to show that it generates exactly L. □

Notice that the proof reveals that the grammar for the string language can be used as a backbone to define the semantics on top.

3 Propositional Logic

Let us start by looking in depth at propositional logic with negation and conjunction. This language will be given almost in the form of an independent grammar, namely by providing an inductive definition. Let $A := \{\top, \bot, (,), \neg, \wedge\}$ be the alphabet, and $E = A^*$, the set of finite strings over A. D is the smallest subset of $A^* \times \{0, 1\}$ such that

- $\langle \top, 1\rangle \in D$.
- $\langle \bot, 0\rangle \in D$.
- If $\langle e, m\rangle \in D$ then $\langle (\frown \neg \frown e \frown), 1 - m\rangle \in D$.
- If $\langle e, m\rangle \in D$ and $\langle e', m'\rangle \in D$ then $\langle (\frown e \frown \wedge \frown e' \frown), \min\{m, m'\}\rangle \in D$.

Let us now define a bigrammar that generates this language. Let $F := \{f_0, f_1, f_2, f_3\}$, $\Omega(f_0) = \Omega(f_1) := 0$, $\Omega(f_2) := 1$ and $\Omega(f_3) := 2$. $\mathcal{I}_b(f_0)() := \langle \bot, 0\rangle$, $\mathcal{I}(f_1)() :=$

$\langle \top, 1 \rangle$. Now, let $n(e) := \texttt{(}\hat{}\neg\hat{}e\hat{}\texttt{)}$. (It is not necessary to restrict the input to a formula since only formulae will be generated anyhow.) Then

(9) $\quad \mathcal{I}_b(f_2)(\langle e, m \rangle) := \langle n(e), 1 - m \rangle$

Likewise, let $c(e, e') := \texttt{(}\hat{}e\hat{}\wedge\hat{}e'\hat{}\texttt{)}$.

(10) $\quad \mathcal{I}_b(f_3)(\langle e, m \rangle, \langle e', m' \rangle) := \langle c(e, e'), \min\{m, m'\} \rangle$

The language that this system of functions generates is exactly D. Notice that all partial functions are even functions.

We now extend this idea to other propositional languages. Suppose that we want to have more basic expressions. Then if they are constants, there is no problem to add them, provided we also name their truth value. So, we can add the following basic sign.

(11) $\quad \langle \texttt{Hesperus is Phosphorus}, 1 \rangle$

On the grammar side all we need to do is add one more 0-ary mode of composition. Or suppose we want to add another connective. If it is truth-functional, again this is straightforward. What is *not* straightforward, though, is the addition of variables. This is where all trouble starts.

Before I start, let me comment on the question of the alphabet. Typically, logic textbooks treat the expression "$\mathtt{p}_i$" as a single letter. Obviously, a language with infinitely many primitive symbols needs infinitely many modes. On the other hand, in actual use the index is produced in decimal notation, so that in fact we do use a finite alphabet. Of course we then have to determine the value of the $\mathtt{p}_i$ starting from a finite list. On that problem see [Kracht, 2011].

First Try. A variable denotes anything that a well-formed formula denotes, so for a variable $\mathtt{p}_i$ we simply enter $\langle \mathtt{p}_i, 0 \rangle$ and $\langle \mathtt{p}_i, 1 \rangle$ into the language. Let D_V be the smallest subset of $(A \cup \{\mathtt{p}_i : i \in \mathbb{N}\})^* \times \{0, 1\}$ satisfying all of the following.

- $\langle \top, 1 \rangle \in D_V$.
- $\langle \bot, 0 \rangle \in D_V$.
- $\langle \mathtt{p}_i, 0 \rangle \in D_V$ $(i \in \mathbb{N})$.
- $\langle \mathtt{p}_i, 1 \rangle \in D_V$ $(i \in \mathbb{N})$.

- If $\langle e, m\rangle \in D_V$ then $\langle (\mathbin{^}\neg\mathbin{^}e\mathbin{^}), 1-m\rangle \in D_V$.
- If $\langle e, m\rangle \in D_V$ and $\langle e', m'\rangle \in D_V$ then $\langle (\mathbin{^}e\mathbin{^}\wedge\mathbin{^}e'\mathbin{^}), \min\{m, m'\}\rangle \in D_V$.

If we want to work with a finite alphabet, we can do the following. Put $B := A \cup \{\mathtt{p}, {}_0, \cdots, {}_9\}$. Now throw out the clauses for the variables and generate the variables step by step. This yields the following grammar.

① $\langle \top, 1\rangle \in D_V$.

② $\langle \bot, 0\rangle \in D_V$.

③ $\langle \mathtt{p}, 1\rangle \in D_V$.

④ $\langle \mathtt{p}, 0\rangle \in D_V$.

⑤ If $\langle e, m\rangle \in D_V$ and e is a variable then for every digit d, $\langle e\mathbin{^}d, m\rangle \in D_V$.

⑥ If $\langle e, m\rangle \in D_V$ then $\langle (\mathbin{^}\neg\mathbin{^}e\mathbin{^}), 1-m\rangle \in D_V$.

⑦ If $\langle e, m\rangle \in D_V$ and $\langle e', m'\rangle \in D_V$ then $\langle (\mathbin{^}e\mathbin{^}\wedge\mathbin{^}e'\mathbin{^}), \min\{m, m'\}\rangle \in D_V$.

e is a **variable** iff it is in $\mathtt{p}({}_0| \cdots |{}_9)^*$. Notice that $\mathtt{p}_0$ is a variable and distinct from $\mathtt{p}$ as well as $\mathtt{p}_{00}$. It is not hard to change this to standard decimal notation but we refrain from going into such detail.

This approach, however, does not work. For it turns out that we can derive both the sign $\langle (\mathtt{p}_0\wedge(\neg\mathtt{p}_0)), 0\rangle$ and $\langle (\mathtt{p}_0\wedge(\neg\mathtt{p}_0)), 1\rangle$. Let us see how this goes. The first is perhaps straightforward:

1. $\langle \mathtt{p}, 0\rangle \in D$. (④)
2. $\langle \mathtt{p}_0, 0\rangle \in D$. (From 1 with ⑤.)
3. $\langle (\neg\mathtt{p}_0), 1\rangle \in D$. (From 2 with ⑥.)
4. $\langle (\mathtt{p}_0\wedge(\neg\mathtt{p}_0)), 0\rangle \in D$. (From 2 and 3 with ⑦.)

However, here is the second derivation.

1. $\langle \mathtt{p}, 0\rangle \in D$. (④)
2. $\langle \mathtt{p}_0, 0\rangle \in D$. (From 1 with ⑤.)
3. $\langle \mathtt{p}, 1\rangle \in D$. (③)

4. $\langle \mathtt{p}_0, 1\rangle \in D$. (From 3 with ⑤.)

5. $\langle \mathtt{(\neg p}_0\mathtt{)}, 1\rangle \in D$. (From 2 with ⑥.)

6. $\langle \mathtt{(p}_0\mathtt{\wedge(\neg p}_0\mathtt{))}, 1\rangle \in D$. (From 4 and 5 with ⑦.)

It is illuminating to see why exactly we get this outcome. Let a_0 through a_9 be new functions on strings, defined by $a_0(e) := e^\frown 0$, $\cdots$, $a_9(e) := e^\frown 9$. All of them are *partial*, being defined if and only if e is a variable, that is, a member of $\mathtt{p}(0|\cdots|9)^*$. Let F_V be $F \cup \{p_0, p_1, g_0, \cdots, g_9\}$ where p_0 and p_1 are zeroary and g_0 through g_9 are unary. They are interpreted as follows.

- $\mathcal{I}_v(p_0)() := \langle \mathtt{p}, 0\rangle$.
- $\mathcal{I}_v(p_1)() := \langle \mathtt{p}, 1\rangle$.
- $\mathcal{I}_v(g_i)(\langle e, m\rangle) := \langle a_i(e), m\rangle$.

This defines an independent bigrammar $\mathcal{I}_v$ in the obvious way (you just need to define $\mathcal{I}_v^\varepsilon$ and $\mathcal{I}_v^\mu$). Each of the two derivations can now be rendered as a constant term. The first is $f_3(g_0(p_0()), f_2(g_0(p_0())))$, or, omitting brackets, $f_3g_0p_0f_2g_0p_0$. The second derivation however comes from the term $f_3(g_0(p_1()), f_2(g_0(p_0())))$, or $f_3g_0p_1f_2g_0p_0$. This term must be different since on the level of generating signs there can be no indeterminacy. Different signs must be generated from different terms.

Thus, allowing variables to denote any truth value is not enough. We must ensure somehow that the values we enter in the derivation are coordinated (to use a term by Kit Fine here, see [Fine, 2007]). Once we enter 1 as value for a variable, we must stick to that value throughout the derivation. But why should that be necessary? If it means one or the other, why can't we choose freely which one it is supposed to mean on each of the occasions? □

Second Try. We give up on truth values. Instead we now claim that wffs denote sets of valuations. A valuation is a function $V \to \{0, 1\}$, where V is the set of variables. Now we define our language D_G as follows.

- $\langle \top, \{0, 1\}^V\rangle \in D_G$.
- $\langle \bot, \varnothing\rangle \in D_G$.
- $\langle \mathtt{p}_i, \{\beta : \beta(\mathtt{p}_i) = 1\}\rangle \in D_G$.

- If $\langle e, m\rangle \in D_G$ then $\langle (\hat{}\neg\hat{}e\hat{}), \{0,1\}^V - m\rangle \in D_G$.
- If $\langle e, m\rangle \in D_G$ and $\langle e', m'\rangle \in D_G$ then $\langle (\hat{}e\hat{}\wedge\hat{}e'\hat{}), m \cap m'\rangle \in D_G$.

Again, to eliminate the infinite set of variables, here is a way to proceed. Given a variable e, let $e^{\S} := \{\beta : \beta(e) = 1\}$. Given a set U of valuations, let $U^{\dagger}$ be that variable e such that $U = \{\beta : \beta(e) = 1\}$. This does not always exist. However, if e is a variable, and $\langle e, m\rangle \in D_G$, then $m^{\dagger}$ exists, is unique and $m^{\dagger} = e$. The signature is as in the previous grammar, except that p_1 is removed from the set of modes. We provide the following interpretation.

- $\mathcal{I}_y(f_0)() := \langle \bot, \varnothing\rangle$.
- $\mathcal{I}_y(f_1)() := \langle \top, \{0,1\}^V\rangle$.
- $\mathcal{I}_y(p_0)() := \langle \mathrm{p}, \mathrm{p}^{\S}\rangle$.
- $\mathcal{I}_y(g_i)(\langle e, m\rangle) := \langle a_i(e), (a_i(m^{\dagger}))^{\S}\rangle$.
- $\mathcal{I}_y(f_2)(\langle e, m\rangle) := \langle n(e), \{0,1\}^V - m\rangle$.
- $\mathcal{I}_y(f_3)(\langle e_0, m_0\rangle, \langle e_1, m_1\rangle) := \langle c(e_0, e_1), \min\{m_0, m_1\}\rangle$.

This works perfectly. The only trouble is that it is hard to see why this is the semantics we had in mind. Why should for example $\top$ denote some set of functions when it actually is thought to denote a truth value?

Such complaints can be countered by saying that our original intuition was unsound. It needed to be revised minimally to make the enterprise work. But more problems arise. The set $\{0,1\}^V$ of all functions from V to $\{0,1\}$ can hardly be said to be a semantic object in the first place, because it presupposes some part of the language to exist independently of the language itself. To see this, recall that a function $\beta : V \to \{0,1\}$ is a set of pairs of the form $\langle v, t\rangle$, where $v \in V$ and $t \in \{0,1\}$. If however this is part of semantics, what is the name of the variable doing here? If $\{0,1\}^V$ could truly be claimed to be the meaning of $\top$, there would be a universal set of variables that we had to claim to exist, and which would be the one that we always had to use when it comes to using variables. This is because the semantic value must be an object which exists independently of the language to be defined. Hence, we must dispense with these functions. □

Third Try. We refine the previous version by introducing possible worlds. We replace the set of functions from V to $\{0,1\}$ by a suitable set W of worlds. Propositions are interpreted as specific subsets of W. This removes reference to variables in the semantics. However, a new problem arises. If we have κ variables but less than 2^κ many worlds we have less worlds than we need to maximally distinguish the values of the variables. That might simply mean that the semantics is chosen incorrectly. But one wonders why the number of possible worlds should have anything to do with the number of *variables* we have in the language. The answer is that this semantics makes the value of a variable a *constant*. Variables and constants are completely indistinguishable. Something is amiss. We shall return to the distinction between variables and constants below in Section 5. □

Fourth Try. I shall outline another account, one that works with finitary valuations (see [Kracht, 2007]). The idea is this. We enter into the language the pairs $\langle \mathtt{p}_i, \{0,1\}\rangle$, for any i. The rule of negation is as follows. If $\langle e, m\rangle$ has been derived, and $m \subseteq \{0,1\}^n$ then $\langle \mathtt{(}\hat{}\neg\hat{}e\hat{}\mathtt{)}, \{0,1\}^n - m\rangle$. This works fine except when $m = \varnothing$, in which case n cannot be chosen uniquely. In this case we simply choose $n := 0$. Conjunction, however, is a hard nut. Let us assume first that e and e' share no variables. Then the new sign is $\langle \mathtt{(}\hat{}e\hat{}\wedge\hat{}e'\hat{}\mathtt{)}, m \times m'\rangle$. In this way, a formula where each variable occurs only once, can be derived. However, notice that the meaning of $\mathtt{(p_0\wedge(\neg p_1))}$ is the same as the meaning of $\mathtt{(p_1\wedge(\neg p_0))}$, or even of any $\mathtt{(p_i\wedge(\neg p_j))}$ namely $\{\langle 1,0\rangle\}$. The association between variables and columns of the vectors is done in the basis of the linear order. Now, when the two formulae do share variables we must look at each of the columns of m and check which ones of m' it must match. To that end we must also know what variable that column represents, and likewise for m'. If they represent the same variable, these columns must be merged. The merger of column i and j in a set of vector results in the subset of all vectors where the ith and the jth column are identical. The net result is that we need many different composition functions for these different cases, and this number is growing with n. Hence this method only works in case n is bounded! □

Fifth Try. There is a somewhat extreme proposal that is inspired by Fine's "coordination", [Fine, 2007]. The idea is that when we have two formulae, conjoining them requires making clear which of the occurrences of a variable of the first formula is to be thought of as an occurrence of the same variable in the second formula, regardless of the name it has been given. This view of coordination

removes the independence of the occurrences in a formula, since rather than by name they have been locked together by construction. Suppose that we deny that in $(\mathtt{p}_0 \wedge (\neg \mathtt{p}_0))$ there are two occurrences of $\mathtt{p}_0$ and instead claim there to be only one. This would then have the favourable consequence that replacement in the sense of the grammar of $\mathtt{p}_0$ by $\mathtt{p}_1$ would yield $(\mathtt{p}_1 \wedge (\neg \mathtt{p}_1))$ since there is only one occurrence that is being replaced. There are many ways in which this proposal can be made to work. One is to differentiate various operations of conjunction. The default operation of conjunction makes all occurrences of variables distinct, say, by suitably renaming the variables of the second formula before concatenation. There are however others that result in making certain variables of the second conjunct identical to the first. For example, if the two conjuncts contain only one variable, then the second operation would make them identical to the first. Syntactically, this requires the following operation c_1. If e contains only one variable p, and e' contains only one variable, p', then $c_1(e, e')$ shall be $(\frown e \frown \wedge \frown e'' \frown)$, where e'' is obtained by string replacement of all string occurrences of p' by p in e'. The formulation of occurrence requires some care in general, but in propositional logic it boils down to mere substring occurrences (see below). The operation is then defined in this particular case. The problem with this attempt is that it cannot work. We need infinitely many operations since the more variables e and e' contain the more possibilities of coordination we need to take care of. The impossibility of such an approach for predicate logic has been demonstrated in [Kracht, 2011], but I guess the proof can be modified for propositional logic as well. □

In order to understand the true predicament, we need to look at the semantics of a variable. As has been noticed over and over, all that matters about variables is whether they are same or different. In and of itself, no variable should essentially have a different semantics from any other variable. Furthermore, according to [Fine, 2007], the meaning of a variable is the entire range of values. Following my own [Kracht, 2011], it might be more appropriate to say that a variable is ambiguous between all possible values that it can assume. Thus in the present context we are led to conclude that the semantics of the First Try is completely right and the repair using valuations or possible worlds is misguided.

Thus, the language for which we ought to give a compositional account is the following language, defined above.

$$D_V := \{\langle e, m\rangle : e \in W, \text{for some } \beta\text{: } m = \beta(e)\} \tag{12}$$

It is embarassing to see the language specified in terms of valuations. It may suffice in order to clearly state what we are aiming for, but it would be discomforting

if reference to valuations were at all necessary in defining the language in the first place. One feels something is fundamentally amiss here.

Hence let me give a somewhat different definition, which will be useful for the discussion in the next section. Say that a **ground substitution** is a function $\sigma : V \to \{\top, \bot\}$. The effect of the ground substitution on a formula is as follows. Recall that a variable p is a string consisting of the letter "$\mathtt{p}$" followed by a decimal string. p is said to **occur** in e if e has a decomposition $e = u^\frown p^\frown v$, where v does not begin with a digit. So, $\mathtt{p}_2$ occurs in $(\mathtt{p}_2\wedge(\neg\mathtt{p}_{17}))$ but not in $(\mathtt{p}_{23}\wedge(\neg\mathtt{p}_{17}))$ since in the latter formula the occurrence ends before a digit (here $_3$). If we have a decomposition $e = u^\frown p^\frown v$ where v does not begin with a digit, the pair $\langle u, v\rangle$ is called an **occurrence** of p in e. Hence, p occurs in e if there is an occurrence of p in e. If q is a formula, replacing the occurrence $\langle u, v\rangle$ of p in e by q gives the string $u^\frown q^\frown v$. q has a new occurrence in the new formula, namely $\langle u, v\rangle$; that occurrence now is no longer an occurrence of p (provided $q \neq p$). Now, $\sigma(e)$ is the result of replacing simultaneously every occurrence of p by $\sigma(p)$, for every $p \in V$.

$$D_V := \{\langle e, m\rangle : e \in W, \text{ there is a ground substitution } \sigma : \langle \sigma(e), m\rangle \in D\} \tag{13}$$

According to Proposition 2 there is a compositional bigrammar for this language. I present such a bigrammar. The set of basic symbols is $\{_0, \cdots, _9, \mathtt{p}, (,), \neg, \wedge\}$. Define the following functions on strings.

$$\begin{aligned}
a_0(e) &:= e^\frown{}_0\\
a_1(e) &:= e^\frown{}_1\\
&\cdots\\
a_9(e) &:= e^\frown{}_9\\
n(e) &:= (^\frown\neg^\frown e^\frown)\\
c_0(e,e') &:= \begin{cases} (^\frown e^\frown\wedge^\frown e'^\frown) & \text{if } \langle (^\frown e^\frown\wedge^\frown e'^\frown), 0\rangle \in D_V\\ \text{undefined} & \text{else.}\end{cases}\\
c_1(e,e') &:= \begin{cases} (^\frown e^\frown\wedge^\frown e'^\frown) & \text{if } \langle (^\frown e^\frown\wedge^\frown e'^\frown), 1\rangle \in D_V\\ \text{undefined} & \text{else.}\end{cases}
\end{aligned} \tag{14}$$

Then we have the following modes: π_0, π_1 (0-ary), $\alpha_0, \cdots, \alpha_9$, ν (unary) and γ_0,

γ_1 (binary). Their action is as follows.

$$
\begin{array}{ll}
\pi_0() & :=\langle \mathtt{p}, 0\rangle \\
\pi_1() & :=\langle \mathtt{p}, 1\rangle \\
\alpha_0(\langle e, m\rangle) & :=\langle a_0(e), m\rangle \\
 & \ldots \\
\alpha_9(\langle e, m\rangle) & :=\langle a_9(e), m\rangle \\
\nu(\langle e, m\rangle) & :=\langle n(e), 1-m\rangle \\
\gamma_0(\langle e, m\rangle, \langle e', m'\rangle) & :=\langle c_0(e, e'), 0\rangle \\
\gamma_1(\langle e, m\rangle, \langle e', m'\rangle) & :=\langle c_1(e, e'), 1\rangle
\end{array}
\tag{15}
$$

It is clear that this bigrammar is independent.

So why isn't this a grammar we can accept? Apart from reservations concerning the generation of variables, which can be dealt with, two problems must be considered. One is the definition of the language, which involves quantifying over substitutions. This is hardly a significant progress over the valuations. It involves quantifying over alternative signs, though ones that do not involve variables. Herein lies the key to our proposal in the next section. The other problem is the definition of the conjunction. One feels betrayed that the reason why the formation of the conjunction of two formulae should fail in one mode when it succeeds in another. And we also do not get an explanation why there is no semantic function of conjunction even used in this definition.

What are aiming for is a formulation of a bigrammar that does not artificially restrict the formation of the conjunction of two formulae. Hence, we want to use only the function $c(e, e')$. That is to say, we have definite intuitions about the syntactic functions that may be used. If that is so, however, it becomes hard to see how a compositional grammar can be given. Indeed, with the functions as above, it cannot exist. For there are formulae e and e' such that $c(e, e')$ can be true while both e and e' can be true (say $\mathtt{p}_0$ and $\mathtt{p}_1$), while there are formulae $\dot{e}$ and $\dot{e}'$ such that $c(\dot{e}, \dot{e}')$ cannot be true while $\dot{e}$ and $\dot{e}'$ can be true (for example $\mathtt{p}_0$ and $(\neg\mathtt{p}_0)$). Hence we need a semantic function w_1 such that $w_1(1, 1) = 1$ and another function w_0 such that $w_0(1, 1) = 0$. This means that we need a mode that combines c on the expressions with w_0 on the meanings ("conjunction to truth") and one combining c on the expressions with w_1 on the meanings ("conjunction to falsity"). However, then we are also able to derive $\langle c(\dot{e}, \dot{e}'), 1\rangle$, which is a contradiction.

Notice that the semantic functions work on impoverished data (only truth values), while the syntactic functions have all information available to them (the language is unambiguous). This asymmetry makes compositionality seem guaranteed.

Let us finally consider a variant of the semantics of D_V. Suppose we put

(16) $e^\circ := \{\beta(e) : \beta : V \to \{0,1\}\}$

Then let

(17) $D_W := \{\langle e, e^\circ\rangle : e \in W\}$

This means that there are now three values: $\{0\}$ ("contradiction"), $\{1\}$ ("tautology") and $\{0,1\}$ ("contingency"). This language is only slightly different from D_V. A tautology now denotes $\{1\}$ in place of only 1, and a contradiction denotes $\{0\}$ in place of only 0. Instead of allowing a contingent formula to have two meanings, 0 and 1, it now has only one meaning, the set $\{0,1\}$. It should be clear that nothing of substance changes had we chosen to use that language instead of D_V. However, I find D_V appropriate.

4 Metavariables

It seems that adding variables makes maintaining compositionality impossible. We are led to conclude that logical languages—in fact *any* language—with variables cannot be compositional. Yet, there appear to be several alternatives on the horizon that are worth exploring. One of them is to actually deny that variables are part of the language.

Recall that in expositions of logic typically two kinds of variables are used: the first are the variables that are part of the language and the second are the variables that are not and are therefore called **metavariables**. Metavariables are placeholders for formulae, while variables of the language are formulae themselves. It is the latter kind of variable that gives us so much headache. The reason is, I guess, pretty simple, and has been pointed out on numerous occasions by Kit Fine: the only thing that distinguishes one variable from the next is its *name*. The semantic difference between variables is therefore nil. Hence, in manipulating variables we must pay attention to their names. This in turn means that we should not even expect a compositional account of them to be viable.

Therefore there seems to be no way of reconciling the idea that variables are part of the language with compositional semantics. However, if they are not part of the language, then we must further ask what logic is all about. The answer that I shall explore here is that logic is about *abstraction* from language. The fact that $(p_0 \wedge (\neg p_0)$ is a contradiction does not lie in the fact that we cannot compositionally generate the value 1 for it. Rather, it lies in the fact that it does not allow to

be instantiated to a true proposition. Therefore, rather than passing immediately to a semantic evaluation we first need to ask what the letter $\mathtt{p}_0$ stands for. Once that is given, a value can be computed. So, only by inserting, say, /`Socrates is wise`/ we find that the sentence can be false. Likewise, to see if it can be true, we need to find a ground instance that denotes 1. Thus, the tautologies of a language are defined with respect to the ground substitutions for that language. It is perhaps instructive to compare Euclid's method of proof with that of Diophantus. Euclid's proofs are modern. They proceed on arbitrary objects. He would start a proof by taking a triangle ABC without specifying the nature of the points A, B and C. Diophantus, on the other hand, always picks concrete numbers and performs the calculations on them. He leaves it to the reader to abstract the recipe from that calculation. His method is only seemingly inferior to that of Euclid, however. Because in both cases we make reference to *actual* objects to substantiate the claims that we are making. The difference is when we want to actually use them in our argument.

Let us pursue this for boolean logic. We start with a language $L \subseteq E \times M$. First we need to define the notion of *substitution instance*. Although clear in the present case, I have argued at length elsewhere that substitution actually presupposes a grammar. This is awkward, but it shows that abstraction is not uniquely determined by the language alone. Consider a grammar over a signature Ω of modes. We define in the usual way terms and functions over the signature. The terms may involve the variables ξ_i, $i \in \omega$, for signs.

It is assumed that terms can be evaluated into a sign $\langle t^\varepsilon, t^\mu \rangle$, using the standard inductive definitions.

Definition 3. *A **schematic expression** is an Ω-term $t(\xi_0, \cdots, \xi_{n-1})$. A schematic expression is **ground** if $n = 0$.*

So far, 'expression' is synonymous with 'term', while 'expression' is otherwise reserved for members of E. The idea is that we can find representatives of the terms that function in the same way as expressions. If t is ground it may be identified with its value under $\mathcal{I}^\varepsilon$ (which is unique, as the language is unambiguous). A schematic expression becomes an expression (via $\mathcal{I}^\varepsilon$) once the variables ξ_i occurring in it are replaced by ground terms. However, in general the result can be rather different depending on what ground expression is being inserted. To see the level of detail needed, consider the expression "$7 + 5$" of ordinary arithmetic. Replacing it for "x" in "$3x$" requires the addition of brackets, so we must write "$3(7 + 5)$" rather than "$37 + 5$". Unless we think of the strings as proxy for the 'correct' notation (see [Machover, 1996], which is representative for such an

approach), the syntax of generating such expressions is somewhat more abstract and means that substitution is more than simple string replacement (see [Kracht, 2004]). I repeat here that substitution is defined only for a given grammar. If we decide to use string replacement for arithmetic terms we end up with a completely different view of abstract truths. There is as yet no reason to suppose that we can exclude this grammar from the range of possibilities.

That said, let us ignore these complications of syntax. I shall write in the usual way "$p_1 \wedge p_0$" to quote the somewhat imperspicuous schematic expression "$c(\xi_0, \xi_1)$" so that in the end we treat the metavariables as if they were genuine expressions (= variables). More concretely, the schematic expression corresponding to a term is obtained in the following way.

$$\begin{array}{ll} (\xi_i)^\spadesuit & :=\mathtt{p}_i \\ (f_0())^\spadesuit & :=\bot \\ (f_1())^\spadesuit & :=\top \\ (f_2(t))^\spadesuit & :=(\mathbin{\hat{}}\neg\mathbin{\hat{}}t^\spadesuit\mathbin{\hat{}}) \\ (f_3(t,u))^\spadesuit & :=(\mathbin{\hat{}}t^\spadesuit\mathbin{\hat{}}\wedge\mathbin{\hat{}}u^\spadesuit\mathbin{\hat{}}) \end{array} \tag{18}$$

(This is based on an infinite alphabet; it is a simple matter to correct that.) Or, more generally, for all f:

$$(f(t_0, \cdots, t_{\Omega(f)-1}))^\spadesuit = f^\varepsilon(t_0^\spadesuit, \cdots, t_{\Omega(f)-1}^\spadesuit) \tag{19}$$

But this obscures the fact that the string functions are actually string polynomials and so the actual form of the translation makes them look like actual formulae with the variable put exactly where the substitution is to be effected.

I shall pass freely between terms and their corresponding schematic expressions.

Definition 4. *Let $\mathcal{I}$ be a grammar for L and t a schematic expression. A* ***ground instance*** *of t is the value $\mathcal{I}^\varepsilon(\sigma(t))$, where σ is a substitution replacing each ξ_i by some ground expression.*

Metatheoretically, we can ask the same questions about substitution for the schematic expressions. Suffice it to say that the replacement must in that case be uniform. Logic arises from a language in the following way.

Definition 5. *Let L be a language with independent grammar $\mathcal{I}$. Then the* ***logical theory*** *of L,* $\mathrm{Th}\, L$*, is that set of schematic expressions t such that for all ground instances u of t, $u^\mu = 1$.*

If the language is ambiguous, the definition is actually more general than might be justified. Namely, given a ground substitution σ, $\sigma(t)^{\mathcal{I}}$ is the sign $\langle(\sigma(t))^{\varepsilon}, (\sigma(t))^{\mu}\rangle$. It may happen that $(\sigma(t))^{\mu} \neq 1$, but that nevertheless $\langle(\sigma(t))^{\varepsilon}, 1\rangle \in D$. This will not happen here, as ground instances have unique values. I have not looked at the complications arising from this exceptional case. Notice also how the definition trades on unique readability.

In general it is necessary to use an equational theory (pairs $t = u$ such that all ground instances receive the same value under all substitutions). But we use the easier case here (benefitting from the fact that we have an algebraizable logic).

It is clear that the logical theory depends on $\mathcal{I}$. Let us take boolean logic, with the grammar given at the beginning of Section 3.

Theorem 6. Th D *under* $\mathcal{I}_b$ *is exactly classical propositional logic.*

Proof. This is actually a trivial consequence of the definitions. Let $t \in \mathrm{Th}\, D$. We have to show that t (under the identification shown above) is a boolean tautology. To this end, choose a valuation β. Now let σ be a ground substitution defined by $\sigma(\xi_i) := \top$ if $\beta(p_i) = 1$ and $\sigma(\xi_i) := \bot$ otherwise. Then $\sigma(t)$ is a ground instance of t and by definition of Th D, $\mathcal{I}(t) = \langle\mathcal{I}^{\varepsilon}(\sigma(t)), 1\rangle \in D$. The value is unique, hence $\mathcal{I}^{\mu}(\sigma(t)) = 1$. It is directly verified that $\mathcal{I}^{\mu}(\sigma(t)) = \beta(t^{\varepsilon})$. Hence, t^{ε} is a boolean tautology. Now let $t \notin \mathrm{Th}\, D$. Then there is a ground instance $\sigma(t)$ such that $\langle\sigma(t)^{\varepsilon}, 0\rangle \in D$. From this we construct a valuation as above such that $\beta(t) = 0$. It follows by similar reasoning that t^{ε} is not a boolean tautology. □

In what is to follow I shall ask how this result can be generalized. Let me highlight a few general facts about this definition. First of all, the set Th D is obviously closed under substitution. For let $t(\xi_0, \cdots, \xi_{n-1}) \in \mathrm{Th}\, D$ and let σ be a substitution of terms $u_i(\xi_0, \cdots, \xi_{m-1})$ for ξ_i ($i < n$). Then every ground instance of $t(u_0(\vec{\xi}), \cdots, u_{n-1}(\vec{\xi}))$ is a ground instance of t, and hence has value 1. Furthermore, Th D is closed under MP (where $\rightarrow$ is either primitive or defined as usual). For suppose that t has the form $u \rightarrow v$ and that both t and u are in Th D. We show that $v \in \mathrm{Th}\, D$. Pick a ground substitution σ. Denote $\sigma(t)^{\varepsilon}$ by t'. Then $t' = u' \rightarrow v'$, where $u' = \sigma(u)^{\varepsilon}$ and $v' = \sigma(v)^{\varepsilon}$. By assumption, $\langle u', 1\rangle \in D$ as well as $\langle u' \rightarrow v', 1\rangle \in D$. Therefore $\langle v', 1\rangle \in D$. The ground substitution was arbitrary. Hence $v \in \mathrm{Th}\, D$.

5 Modal Logic

One of the attempts to salvage compositionality was the introduction of possible worlds. I have criticised that proposal for blurring the distinction between variables and constants. I shall show in some detail here how that same confusion affects modal logic itself. We shall see that many standard systems trivialise under the approach sketched here if they have no constants.

That said, let us now turn to modal logic. The definitions remain in place. However, the semantics is somewhat more involved. To see the difference with boolean logic notice that in boolean logic there are two possibilities: a proposition is true, or it is false. In modal logic, however, there are many more truth values. In fact, definitions of logics usually start with an intuition about what the proper axioms are and then ask how the semantics fits in. In the scheme of things exposed here this cannot be done. We must first determine what the values are, because the language pairs expressions with meanings so that we must first say what the meanings are.

Let $\mathfrak{F} := \langle F, R, U\rangle$ be a general frame. This means that F is a set (the set of possible worlds), $R \subseteq F^2$ the so-called **accessibility relation** on F, and $U \subseteq \wp(F)$ a set of subsets of F closed under intersection, relative complement and

$$\tau\, a := \{w : \text{ for all } v\text{: } w\, R\, v \Rightarrow v \in a\} \tag{20}$$

We put $M := U$ and define the language $D_{\mathfrak{F}}$ as follows. (Actually, we define an independent bigrammar that generates this language.) The grammar $\mathcal{I}_b$ is extended by another unary mode, m, to yield the grammar $\mathcal{I}_m$. We put

$$\mathcal{I}_m(m)(\langle e, m'\rangle) := \langle (\hat{}\,\Box\,\hat{}\,e\,\hat{}), \tau\, m'\rangle \tag{21}$$

This is clearly independent. Finally, we also define the function $(-)^{\spadesuit}$ from schematic expressions to terms, by adding one clause to (18):

$$(m(u))^{\spadesuit} = (\hat{}\,\Box\,\hat{}\,u^{\spadesuit}\,\hat{}) \tag{22}$$

Recall that given a modal logic Λ, we define the canonical frame $\mathfrak{Can}_\Lambda(V)$ over a set V of variables as follows. Let L_V denote the set of formulae built from V with the functions f^ε. W_V is the set of all maximal consistent sets of formulae from L_V. For any two such sets G and H, we put $G\, R_V\, H$ iff for all $(\hat{}\,\Box\,\hat{}\,e\,\hat{}) \in G$ we have $e \in H$. Finally, for a formula $e \in L_V$, let $\hat{e}_V := \{G \in W_V : e \in G\}$, and $U_V := \{\hat{e}_V : e \in L_V\}$. Then

$$\mathfrak{Can}_\Lambda(V) := \langle W_V, R_V, U_V\rangle \tag{23}$$

The structure of $\mathfrak{Can}_\Lambda(V)$ depends only on the cardinality of V (apart from Λ of course). So one usually writes $\mathfrak{Can}_\Lambda(\kappa)$, κ a cardinal number.

Now let Λ be given. As before, if the semantics is to be compositional we are not allowed to have any variables. So, the correct semantics is defined by the frame $\mathfrak{Can}_\Lambda(\varnothing)$ and nothing more. This means that the logic Λ gives rise to the following language.

Definition 7. *Let Λ be a modal logic. The **canonical language** associated with Λ is $\{\langle e, \hat{e}_\varnothing\rangle : e \in L_\varnothing\}$. It is denoted by* $\mathrm{Lgc}(\Lambda)$.

To see where this leads us we give a concrete example. Given Λ, write $\varphi \equiv_\Lambda \chi$ if both $\varphi \to \chi \in \Lambda$ and $\chi \to \varphi \in \Lambda$, using $\varphi \to \chi$ as an abbreviation for $\neg(\varphi \wedge \neg\chi)$. Consider the modal logic D, which is K plus the axiom $\Diamond\top$. In D, there are only two constant formulae, since $\Diamond\top \equiv_\mathsf{D} \top$, $\Diamond\bot \equiv_\mathsf{D} \bot$. (In K there are infinitely many nonequivalent constant formulae.) Thus, if a formula is a theorem if every variable free instance is we conclude that every formula is either a theorem or its negation is. There are up to equivalence only two constant formulae. This means that the canonical language is the following:

$$\mathrm{Lgc}(\mathsf{D}) = \{\langle e, \varnothing\rangle : e \equiv_\mathsf{D} \bot\} \cup \{\langle e, W_\varnothing\rangle : e \equiv_\mathsf{D} \top\} \tag{24}$$

Let us now see what happens if we abstract the logic from this semantics. We have just seen that there are no contingent formulae. Furthermore, $\Diamond\Diamond p \to \Diamond p$ turns out to be a theorem: suppose we substitute $\top$ for p. Then since $\Diamond\top \equiv_\mathsf{D} \top$ we have $\Diamond\Diamond p \to \Diamond p \equiv_\mathsf{D} \top$. Suppose we substitute $\bot$. Then since $\Diamond\Diamond\bot \equiv_\mathsf{D} \bot$, $\Diamond\Diamond p \to \Diamond p \equiv_\mathsf{D} \top$. This shows that $\Diamond\Diamond p \to \Diamond p$ is a theorem of $\mathrm{Lgc}(\mathsf{D})$. Likewise one can show that $\Diamond p \leftrightarrow p \in \mathrm{Lgc}(\mathsf{D})$. Recall that $\mathsf{K} \oplus p \leftrightarrow \Diamond p$ is the logic of the one point reflexive frame, and that it has only two extensions: itself and the inconsistent logic. This means that all consistent logics above D collapse into one. To put this into perspective, define the following.

Definition 8. *Let Λ be a modal logic. Then $\mathsf{Th}\,\mathfrak{Can}_\Lambda(\varnothing)$ is called the **substitutional companion** of Λ and denoted by* $\mathrm{Cp}(\Lambda)$.

To see the origin of the nomenclature, let us give an alternative characterization.

Proposition 9. $\mathrm{Cp}(\Lambda)$ *is the set of all φ such that for all ground substitutions σ $\sigma(\varphi) \in \Lambda$.*

Proposition 10. *The substitutional companion of Λ is the logical theory of its canonical language.*

Clearly, Λ is contained in its substitutional companion. For $\Lambda = \mathrm{Th}(\mathfrak{Can}_\Lambda(\omega))$, and $\mathrm{Th}(\mathfrak{Can}_\Lambda(\kappa)) \subseteq \mathrm{Th}(\mathfrak{Can}_\Lambda(\lambda))$ whenever $\lambda \leqslant \kappa$.

The difference between these logics can be rather large, however. From what we observed above, the 0-generated canonical frame consists of a single world. Hence we obtain

Proposition 11. *Let $\Lambda \supseteq \mathsf{D}$ be consistent. Then the substitutional companion of Λ is $\mathsf{K} \oplus \Diamond p \leftrightarrow p$.*

Is that at all a reasonable outcome? I think yes. For all the talk of arbitrary propositions hides the fact that they do not necessarily do us any service. If all we can really express is $\top$ and $\bot$, talk of an arbitrary constant is meaningless. How we are going to justify a weaker logic if we cannot supply any counterexample to the stronger postulates? In a way, then, being able to express less means that your logic grows in strength since there are less possibilities to disprove some formula.

Thus, in order to get more discriminatory power we must introduce more constant propositions. Essentially, as the logic is defined by its countably generated canonical frame, we get our logic back provided we have enough constant propositions. Depending on the logic itself, "enough" may mean some finite number or $\aleph_0$. Let us take a closer look.

We first assume that we already have an expansion L_C of our language by some set C of constants. Generally, these constants need not be *independent*. By that we mean that the logic in the language expanded with these constants is the substitutional closure of Λ in the new language. Or alternatively, that the logic in the language with the constants is axiomatized by Λ (though it is not identical with Λ). This is stronger than mere conservativity which only requires that adding the constant does not add tautologies in the old language. For example, expanding D by some constant c may lead to a logic in which $c \leftrightarrow \top$ is valid. This may well occur. Since this is not the result of substituting for a formula, the constant is not independent. So, to determine the fate of our constants we look at the logical theory of two languages: the one defined from the expanded language L_C and the other defined from the unexpanded language L. Call the first T_C and the second T. Now let $T' := T_C \cap L$. This is the subset of T_C containing all formulae from L. Clearly, $T' \subseteq T$, for L_C allows more ground instances than L. It may also happen that $T' = T$. If we add countably many independent constants we get back our original logic. This is summarised in the next theorem.

Theorem 12. *Let L be a string language. Let C be a countably infinite set of independent propositional constants and denote by L_C the expansion of L by C.*

Let Λ be a modal logic over L, and Λ_C the minimal extension of Λ in the language L_C. Then $\Lambda = \mathrm{Cp}(\Lambda_C) \cap L$. Moreover, $\mathrm{Cp}(\Lambda_C)$ is the result of closing Λ substitutionally in L_C.

6 Predicate Logic

Having shown how we can find a semantics for modal logic let us now briefly look at predicate logic. Predicate logic, it turns out, is markedly different from modal logic. In modal logic the countably freely generated frame is a generic semantics, and we can restrict ourselves even to a specific valuation. Similarly in propositional logic, where a single matrix is typically enough. Everything can be decided by looking at a specific model. In predicate logic this is the exception rather than the rule. Unless the theory is categorical in some infinite cardinality there is no hope of using a single model. Unlike modal logic, moreover, the quantifiers are part of the language, and thus the elimination of variables is not likely to proceed. Kit Fine has made an attempt to overcome the situation using arbitrary objects ([Fine, 1985; Fine, 1983]). Arbitrary objects are like Platonic ideas. When in a proof you consider an arbitrary triangle and subsequently perform operations on it, you are not doing it on a specific triangle, rather on an "ideal" version of it. The problem with this is that we shall need to give up classical logic. For given an arbitrary triangle, neither can we say that it is isosceles nor can we say that it isn't. This is the same when we consider abstract objects in general. What it comes down to—in the end—is that we can only say that an arbitrary triangle has some property if all its instances have that property. Which means that again we must work our way up from the basic instances.

Let us see how that goes. Consider adding a countable set C of constants. Assume that there are no function symbols, only relation symbols. Furthermore, let equality not be numerical identity but simply be an equivalence relation. Finally, assume that the structure contains only those elements that are denoted by some constant. (This is something of a covering rule. This can be assumed due to the Löwenheim-Skolem Theorem without affecting the logic.) Now the quantifiers can be seen as proxy for large disjunctions and conjunctions as follows.

$$(25) \qquad ((\forall x)\varphi)^\heartsuit := \bigwedge_{c\in C} \varphi^\heartsuit[c/x], \qquad ((\exists x)\varphi)^\heartsuit := \bigvee_{c\in C} \varphi^\heartsuit[c/x].$$

(Otherwise, $(-)^\heartsuit$ commutes with conjunction and negation. For a prime formula $\varphi^\heartsuit = \varphi$.) This translation totally eliminates quantifiers and variables. Models

can be built over the set C in the obvious way. However, we still need to face up to the fact that there are several nonisomorphic models. So, our semantics is rather a S5-frame with constant domain C. Again, for each individual structure we know whether or not a given constant formula is true. Finally, we accept a predicate logical formula φ as true if $\varphi^{\heartsuit}$ is true in every structure, or if $\Box\varphi^{\heartsuit}$ is true simpliciter at one (in fact, any) world.

7 Conclusion

The idea that logical languages must be compositional has been closely examined. The conclusion is that we should as a rule rather expect that logical languages are *not* compositional. And that has a reason worth exploring. Logical formulae do not come primarily with a direct semantics. Logic comes after we have settled on a language, which is defined to be a relation between expressions and meanings. Logic then analyses the relationship between expressions in terms of their truth. When it uses variables, however, it does not make them figure as genuine objects of the language (= formulae) but rather *always* thinks of them as schematic. The difference is easily missed. To give an example, think of defining some modal logic. A particular formula is a tautology as long as it is true for every concrete proposition we can put in place of the propositional variables. In terms of possible worlds, the constant propositions take values that are *absolutely definable* sets of worlds. (They can be defined without knowing the valuation.) In a nonschematic understanding, however, we may have to consider also those sets that are not definable by constant propositions. However, it is hard to see how such propositions can figure in the definition of the logic itself. How do you effectively show that a particular argument is invalid by pointing at some counterexample that you cannot properly construct? The construction cannot be effected within the language itself, it must rely on a higher order language within which the reasoning can be couched.

This is not to say that there can be no general truth expressed within the language itself. Generality is not what prevents a language from being compositional. However, schematicity is. It transcends language in order to discover general patterns of inference within it. We have shown that mostly variables can only be understood as schematic. This is in sharp contrast to natural languages, which have no variables and therefore can rely on algorithms of meaning composition that are not schematic.

References

[Fine, 1983] Kit Fine. *A Defense for Arbitrary Objects*. Number LXVII in Proc. of the Aristotelian Society. 1983.

[Fine, 1985] Kit Fine. Arbitrary objects and natural deduction. *Journal of Philosophical Logic*, 14, 1985.

[Fine, 2007] Kit Fine. *Semantic relationism*. Blackwell, London, 2007.

[Kracht, 2004] Marcus Kracht. Notes on Substitution in First–Order Logic. In Uwe Scheffler, editor, *First-Order Logic 75*, pages 155–172, Berlin, 2004. Logos Verlag.

[Kracht, 2007] Marcus Kracht. The Emergence of Syntactic Structure. *Linguistics and Philosophy*, 30:47–95, 2007.

[Kracht, 2011] Marcus Kracht. *Interpreted Languages and Compositionality*. Number 89 in Studies in Linguistics and Philosophy. Springer, 2011.

[Machover, 1996] Moshé Machover. *Set theory, logic and their limitations*. Cambridge University Press, Cambridge, 1996.