Typed and Untyped Terms in Compositional Semantics

N. J. H. Wontner

199
Thesis

Faculty of Philosophy
University of Oxford
April 2018

Abstract

Categorical grammars can be used to underpin computationally effective compositional semantics, since they do not rely on semantic filtering. A derivation rule on a small set of 'basic' types provide a rich landscapes of types.

In simple accounts, each lexeme from a language has a single type, based on their semantic category. However, natural languages are less rigid in terms of category combinations. In order to maintain compositional simplicity, lexical units in natural languages are best analysed as having different types in different expressions, yielding type ambiguity. Classic examples are the logical connectives. For example, **and** can be used to combine sentences, noun phrases (type e), verb phrases (type (e, t)), etc. We investigate these cases, and the prevalence of plausibly multi-typed lexemes in natural language, outside of connectives.

We explore the standard flexible total type approaches: stipulative and rule-generated homophone theories, and rule-based type shifting. Both are found wanting. Issues include deviation from competent speaker intuitions, lack of theoretic economy, and arbitrary assignment of type to terms which are not obviously typed.

Finally, we sketch a variant on Montague grammar, Mix, where some lexemes are 'untyped'. After giving independent definitions for the untyped lexemes, we argue that Mix is no more procedurally inadequate or in need of semantic filtering than the alternative accounts. In particular, any calculation meta-rule is no stronger than the implicit meta-rules of the alternatives. Finally we argue that Mix avoids arbitrary type assignments, is theoretically parsimonious, and is conceptually independent. It possibly even aligns with speaker intuitions.

Acknowledgements

I would like to thank my supervisor, Prof. Paul Elbourne, for his guidance and criticism throughout this work. I am very grateful to Prof. James Studd for his formative teaching, and influence on my critical approach. My thanks extend to Prof. Lucas Champollion (NYU) for his correspondence, and James Kirkpatrick for some helpful conversations. I thank Eva Levelt for her time and comments, and my parents, for their love and support. My greatest gratitude extends to Zoë Chatfield, for continuing engagement, and her advice on all matters linguistic. I thank every person who, during the writing process, has answered the question "Is this a natural sentence?". Finally, I would like to thank my outstanding friends and colleagues, who continue to be interested and encouraging in such plays of passion.

Contents

1 Historical Context

1.1 Function-Argument Semantics

We are investigating meaning, specifically how we can generate the meaning of a composite expression from the meanings of its subexpressions. Guiding the tradition of formal semantics is the Principle of Compositionality[1]:

> "The meaning of an expression is a function of the meanings of its parts and the way they are syntactically combined."
>
> *Partee 1984*

A traditional account builds a formal system of *semantic types*, and then supposes that *each and every* expression has a particular type. A system might contain rules for how to 'derive' new types, or how to 'apply' an expression of a certain type to another to calculate composite meanings.

We are principally concerned with a possible misstep amongst the contemporary accounts: might our best formal semantics for natural language have some *untyped* lexemes?

To unpack exactly what this might mean, we review some relevant groundwork in formal semantics. Building on the work of Frege, Tarski, Kripke, etc., *function-argument* (or model-theoretic) semantics is the standard approach in formal semantics today. Two basic Fregean ideas are at its core[2]:

1. We analyse expressions into a main functor and its argument(s).
2. We distinguish categories of functors according to the categories of their arguments and values.

We are principally concerned with 2. Cresswell summarises what philosophers require from a formal semantics for natural language:

> "what is usually important to [philosophers] is to see how an expression can be produced in a formal language, with a precise model-theoretic semantics, which contains symbols which mimic the words of natural language sentences"
>
> *Cresswell 1977*

However, modern Montague grammars[3] seem to assign implausible, arbitrary types to certain lexemes of natural language. We suggest that these lexemes are untyped, and governed by their own unique semantics.

[1] Jacobson 2001.
[2] Geach 1970
[3] Montague 1970(a), 1970(b), 1973

We present precise model-theoretic semantics which is a variant/extension of standard Montagovian type theory, but is only *partially* typed. It includes a special class of expressions, the *untyped* terms, whose meaning and interactions with the typed terms are explicitly characterised.

1.2 Motivating the Theory of Types

Why should we type at all? Why not leave all arguments and functions untyped, and have a totally general application rule which permits application (i.e. substituting a term for a variable) of any function to any argument?

We motivate typing on three fronts.

1. Adequate Compositional Semantics of Natural Language

   Standard predicate logic is inadequate as a semantics for natural language, e.g. it is only powerful enough to talk about the properties and relation of entities. Type theory *extends* these basic logics[4], and better explains our natural language data[5].

   > "type theory is so much more similar to language than predicate logic is, that adopting it as a vehicle of representation can overcome the mismatches between grammatical form and predicate logical form"
   >
   > *Muskens 2011*

2. Computational/Procedural Effectiveness

   Typed accounts are highly algorithmic[6] (see § 4.4). That is, there is a determinate way to calculate the meaning of composite expressions given only the rules of the system and the meanings of their subexpressions.

3. Intractable Paradoxes

   Certain paradoxes result from having *totally untyped* semantic languages, particularly Russell's paradox and the Liar.

We outline the paradoxes of 3. and their (relatively isolated) typed theoretic solutions. We then progress to 1., 2. and the major discussion of semantic type theory.

1.3 Russell's Theory of Types

1.3.1 Russell and the Liar

Russell first proposed his theory of types[7] as a solution to several paradoxes concerning self-reference and "totalities", including the Liar and the Russell set: $P_\perp := \{x : x \notin x\}$.

[4] GAMUT 1991
[5] Lewis 1972
[6] Moortgat 2002
[7] Russell 1908

Each paradox was linked to the pre-theoretic expressive power of natural languages. The paradoxical expressions seemed well-formed, and so were assumed meaningful. However, they had no adequate meaning. Russell's solution effectively rejected the assumption that these were meaningful under their natural interpretation.

To avoid Russell's paradox, each mathematical object was assigned a type. Objects of type k were built only from objects of type $< k$, so *not* from objects of their own type: "whatever involves all of a collection must not be one of the collection"[8]. This generated an ordered hierarchy of mathematical objects.

To solve the paradox, note that any set S is constructed from some objects (sets) with strictly lower types. Variables in the definition of S do not range over S itself. So, $P_\perp$ is not a well formed definition. The closest legitimate definition is P_k, of type k, composed of all and only those objects *of type* $< k$ which are not members of themselves. P_k is not one of these. So there is a determinate answer, $P_k \notin P_k$.

Russell also details how we generalise the theory to resolve other self-referential paradoxes, e.g. the Liar, λ, which says that λ is not true:

$$\lambda = \neg Tr \ulcorner \lambda \urcorner$$

The solution is exactly parallel: sentences are constructed from expressions of a lower order. Tarski developed this idea[9], positing a hierarchy of the *semantic predicates*, Tr_k. If $k \leq l$, and a sentence, ϕ, is type $m < k$, then Tr_l acts like 'truth' on ϕ. In jargon, Tr_l is truth-like[10] on ϕ. If a sentence, ϕ, features a predicate of type k, ϕ is of type (at least) $k+1$. So the truth predicate Tr in λ was of some type $k \leq l$ where λ is type l. But then Tr_k is not truth-like[11] for λ. This resolves the paradox.

Here we see (ramified) type theory moving from a mathematical setting to a semantic setting, where sentences *and* semantic predicates are typed.

1.3.2 A Taste of Things to Come

These thoughts anticipated the foundations of modern formal semantics in the '60s and '70s. However, greater generality was needed. Compositional semantics requires much more typing than just typing the semantic predicates. Canonically, the *whole* lexicon is typed, so-called *total* typing.

[8] Russell 1908

[9] Tarski 1944

[10] Russell-Tarski sentences are *type-constant*: for a sentence, $\mathbf{true}_k \rightarrow (\forall l < k\ \mathbf{true}_l)$. So each sentence is uniquely identified by its *least* type. So, given different semantics, this multi-typing reduces to mono-typing (see § 4.5).

[11] These 'type-mismatches' anticipate modern accounts of grammaticality *without* semantic filtering (§ 1.4).

So far, the only typed expressions are sentences and semantic predicates. Theories of *truth* typically only have *sentences* as arguments, yielding a simple hierarchy. Whereas formal semantics for natural languages must explain a greater variety of expressions.

Already in these early type theories, purportedly individual lexical units were being given complex analyses. The English word **true** was analysed as ambiguously denoting many different predicate homonyms. Stronger still: the intuition that there is a single *absolute* truth predicate is *false*; no Tarkian predicate is sufficiently general.

1.3.3 Superseding Type Solutions: ZFC

Russell's type theory is *not* the contemporary canonical solution to Russell's paradox. This marks the beginning of a trend; whereby historic type theoretic solutions are superseded.

Instead, mathematicians use a more sophisticated set theory, ZFC[12]. Russell's paradox is resolved by restricting the separation axioms:

Separation. *For any set x, any predicate (in the logical language) ϕ,*

$$\exists z\ \forall y\ ((y \in x \wedge \phi(y)) \Rightarrow y \in z)$$

Properties only legitimately separate a *subset* from an other*, 'pre-existing'* set. In ZFC, Russell's argument is not paradoxical, instead it proves that there is no universal set. Crucially, there is no *overt* hierarchy of mathematical objects, i.e. *no typing*.

Even so, a standard metaphysics of set, the Iterative Conception[13], is type-theoretic in flavour. Roughly, one conceives of sets being *constructed* in stages. There are some initial sets. Then the axioms legitimate certain constructions of 'second layer' sets. Iterating this process generates a hierarchy. These stages seemingly correspond to types: the 'type' is the ordinal which indexes the stage at which the set appears[14].

1.3.4 Superseding Type Solutions: Kripke

The story for the Liar paradox is similar. Russell solves the paradox by typing the language's truth predicates and sentences. This traditional account has *also* been challenged, by Kripke's solution[15].

Rather than having several semantic predicates, Kripke has a *single*, untyped, semantic predicate. In brief, he entertains the hierarchy, expanding the

[12] Kunen 2011

[13] Boolos 1971

[14] Axiomatic set theory explicitly use types (*ranks*, in Kunen 2011). New Foundations is also typed (Bowler & Forster 2017).

[15] Kripke 1975

extension and anti-extension of his predicate **true** to include every sentence whose truth-value has been 'determined' at that level of the hierarchy. The resultant (paracomplete) account has a *single* truth predicate, with no hierarchy of sentences, so no implicit typing.

(Total) type theories have been used to solve philosophical problems in mathematics and logic. In both cases, non-typed accounts have superseded them. This gives us reason to be cautious about typing in formal semantics.

1.4 Foundations of Modern Formal Semantics

Lewis, Cresswell, and especially Montague expanded typed analyses much further, to provide a compositional account of semantics for formal and natural languages.

The overarching idea is that just as linguistic expressions belong to different syntactic sorts (possibly 'folk grammatical' sorts e.g. nouns and adjective, or something more theoretical), so too their meanings belong to various *semantic types*. The claim is that these are correlated: "interpretations of different expressions of the same category c belong to the same semantic types."[16].

The semantics are based on functions and arguments. The same term might appear as an argument in one context and as a function in another. If the argument is 'appropriate' for the function, an application rule can determine the meaning of the composite expression from the argument and function subexpressions. For example:

(1) Mary walks.

Here, **walks** is a function, being truth-evaluable on a certain class of arguments (entities) and not meaningful with other arguments. **Mary** is the argument. We know the meaning of **Mary**, namely the person Mary, and the meaning of **walks**, which (extensionally) is a function from objects to truth-values. We can compose these meanings using an application rule to generate the denotation of the whole expression, (plausibly) a truth-value.

However, in other composite expressions, the arguments and functions 'do not fit', so we cannot calculate a meaning. For example:

(2) Loves walks.

Even given the meanings of **loves** and **walks**, we cannot calculate a meaning for the entire expression. Intuitively, **loves** is not the right *kind* of argument for **walks**.

By *typing* the expressions, we do not rely on any intuition to calculate the meaning (or lack thereof) of the composite expression.

[16] Hendriks 1993

So, this system does not require any *semantic filtering*[17]. One could write down instructions (for a computer, or someone who does not understand the language) on combining the meanings of expressions of various types to generate the meaning of compound expressions. These calculations *do not* require any prior understanding of the meanings of the terms, they are *purely algorithmic*.

Suppose we are given the expression:

(3) Gramble spunt.

Given that **Gramble** is type x and **spunt** is type (x, t), we can use the composition rules to generate the meaning of the full expression from its constituents. The types and meanings of its subexpressions suffice to generate this novel meaning.

1.5 A simple typed language, Basic

In the spirit of Montague, Lewis, and Cresswell, we define a system of *semantic categories*, S. If lexemes have the same type, their lexical entries are semantically alike. The semantic analysis herein is *simple*, i.e. each lexeme is assigned a single type.

We start with a collection of *foundational* types.

Basic 1 (Ontology)**.** *The foundational types of Basic are $e \in S$, for entities, and $t \in S$ for truth-values.*

Some expressions have these types, indicating *what kind* of meaning they have. Having type e corresponds to the expression denoting an entity, which might suit a proper name.

From the basic types, one can derive[18] further *functional* types.

Basic 2 (Derivation)**.** *If $c, d \in S$ (i.e. are types), $(c, d) \in S$.*

An expression has type (c, d) when it denotes a function whose *arguments* are type c and *values* are type d. For example, predicates of type (e, t) denote functions from entities to truth-values.

Finally, S is defined as the smallest system closed under these rules.

Basic 3. *If $(c, d) \in S$ then c, $d \in S$.*

We then define the domain of the types. Suppose D is a collection of entities.

Basic 4 (Semantic Denotation Domain)**.**
$\boldsymbol{D}_{e,D} = D$
$\boldsymbol{D}_{t,D} = \{0, 1\}$
If c, d are types: $\boldsymbol{D}_{(c,d),D} = \boldsymbol{D}_{d,D}^{\boldsymbol{D}_{c,D}}$

[17] See Ladusaw 1986. Here, semantic well-formedness only depends on the types of the subexpressions, judging it requires *no lexical understanding*.

[18] Some grammars derive *bi-directionally* (Lambek 1999). We assume with Lewis that a transformational component suffices, see § 1.6.2.

The meaning of a lexeme is an entity from D (e.g. the proper noun **Mary**) precisely when it is type e. Exactly similarly, "the denotation of a transitive verb like **like** is a *function from individuals to functions from individuals to truth values*"[19], i.e. type $(e,(e,t))$.

These domains are largely implicit, as we focus on particular concrete definitions. We work in an extensional account for simplicity, but this can be generalised to include intensionality[20] or event semantics.

This describes a framework for *lexical* semantics, but says nothing of composition. Montague claimed that each semantic category has its own unique composition rules[21]. But this is unnecessarily theoretically complex, and flouts the expected generality of rules of natural systems.

Instead, a few reasonable assumptions about the syntax-semantic interface[22] yield a more general compositional system. We assume that syntactic trees are binary branching, and that semantic interpretation rules are local (the denotation of a node is computed from the denotations of its daughter nodes only). Most importantly:

Frege's Conjecture. *Semantic composition is functional composition.*

So, Basic has a single (functional-)application rule:

Basic 5 (Left Application)**.** *An expression of type c applied to a type (c,d) yields type d like so: $c \cdot (c,d) \mapsto d$.*

Derivation and Application together constitute a natural many-sorted algebra on the collection of types: we can derive type (c,d) from types c and d then left-apply c to return to d[23]. These form a basic categorical grammar, with simple and elegant rules[24]. The two rules act naturally on the syntax *and* semantics of the language, so we can justifiably think of the types as the *semantic* category of the expression. We call a series of uses of the application rule on an expression a (semantic) *calculation.*

> "a [calculation] encodes an effective procedure for building up the structural organization of an expression, and for associating this structure with a recipe for meaning assembly"
>
> *Moortgat 2002*

[19] Heim & Kratzer (now H &K) 1998
[20] GAMUT 1991
[21] Montague 1970(b)
[22] H & K 1998
[23] Hendriks 1993
[24] Lewis 1972

1.6 Preliminary Issues

The account so far is inappropriate for natural language semantics. Lewis points out two faults.

1.6.1 Word Order

Firstly, there is the issue of word order. We can tease this out via right application or via non-adjacency.

We seem to need *right* application, as well as left.

(4) He quickly ate the cake.

Here, the verb **ate** is type (e, t), but is modified[25] by an adverb **quickly**, $((e, t), (e, t))$, which lies on its *left*:

$$((e, t), (e, t)) \cdot (e, t) \mapsto (e, t)$$

However, this is not justified by left application, so we need a right application rule to cover these instances[26].

There are also problems when the words are not precisely adjacent in the expression. For example:

(5) Jill, who's over there, skis.

The inserted clause seems only to provide contextual information. In which case, we want to apply **Jill** directly to **skis**, which are non-adjacent.

1.6.2 Iterated Modifiers

There are also *inappropriate* iterations of the derivation rule. We assume a principle of axiomatic naturalism: nature generally follows rules. So, as a natural system, we expect language to follow rules[27]. Our aim is to find a *concise* expression of the (genuine) rules of semantics. However, it seems the system so far has *too many* types to be natural.

Lewis is apparently concerned with 'small' derived type[28], like $(e, (e, (e, (e, e))))$. Types are (supposedly) *natural semantic categories*, yet these are never witnessed. Even if we hope to provide *unified* semantics, which adequately explain any language, many types generated by a totally general derivation rule are never used.

Pushing this further, the system allows for *infinitely* many different types. However, it seems that the number of types in natural language is finite[29]. In

[25] Possibly, see § 2.3.
[26] Bar-Hillel 1964
[27] Wigner 1960
[28] Lewis 1972 §2
[29] English has "unbounded novelty" when presented recursively, but "the set of all and only the strings of English... is not a natural phenomenon" (Collins 2010), the naturally occurring part is only finite.

fact, the number of *lexical units* in natural language is seemingly finite. So a totally general derivation rule is overly strong.

Lewis claims that we can solve these problems with a "*categorical-based* transformational grammar"[30]. This system analyses "John Mary loves." as an ordinary, meaningful sentence of English (§ 1.6.1). However, the word order can be rectified by a transformational component[31]. We assume that some suitable (Chomskyan) transformational component settles word order issues, for a suitable base component.

It is unclear how transformational methods could alleviate the difficulties of iterated modifiers. One alternative solution is to stipulate an upper limit to the number of iterations, another is to disallow certain uses of Application. These stipulations might be informed by natural language. In fact, they might be serious constraints: we rarely need categories which cannot be derived in fewer than four steps, nor are there *any* clear English examples of beyond ten steps. In § 3.1 and § 3.2, we explore disadvantages of stipulating such limits on the rules of a system.

1.7 Simple Typing

Basic 6 (Right Application)**.** *An expression of type c can be applied to a type (c, d) to yield an expression of type d like so:* $(c, d) \cdot c \mapsto d$

Let's take the typed system Basic to consist of the ontology and the three rules above, along with some *appropriate* transformational structure. We abbreviate left and right application as **(A)**. Here, 'appropriate' means the kinds of transformations that might be expected in a Chomskyan transformational generative grammar[32]. We don't stipulate any precise limits: the linguists can decide the details[33]. Finally, let each and every lexical entry for a lexeme be assigned a single type (i.e. Basic is *simply* typed).

The categorical grammars build on this form, becoming ever more sophisticated and nuanced, to cope with various linguistic phenomena in several natural languages. The literature is rich, for example Lewis explores categorical explanations of non-declarative sentences.

[30] Presumably Lewis means (fairly weak) early-Chomskyan transforms, e.g. transforming a declarative sentence into a question.

[31] Lewis claims that "The phrase-structure rules are implicit in the system", e.g. $S \mapsto NP.VP$. But this is apparently limited to *left* rules.

Instead, *antisymmetry* asserts that (syntactic) tree structure dictates word order (Kayne 1994). Here, the hierarchical structure corresponds to a *unique* linearisation (specifier-head-complement branch order), solving the word order issue.

[32] Chomsky 1965

[33] A semantic theory requires a *syntactic base component*: some phrase structure rules which produce some basic structures, which are then transformed (Chomsky 1965). The rules of Basic have syntactic effects, so we may not need much of a base component.

1.8 Flexible Typing and Type Ambiguity

In Basic, *each and every* lexical unit has exactly one type. This lends the account considerable elegance and simplicity. It makes the type-assignment map a genuine function. It also seems to align with intuitions, (naïve) canonical definitions are only ever *of one type*: **Fido** denotes *only* an entity, and so is type e[34], and **barks** *only* evaluates entities to truth-values.

More convincingly, simply typed systems are somehow *complete*: **(A)** corresponds to the 'reverse' of derivation, so any step is reversible. Having derived (c, d) from categories c and d, application by category c yields the original category d.

Formally, the algebraic structure is roughly a pre-group[35]: it is closed under adjoints and multiplication, but has no 'identity' element (i.e. no type i such that $c \cdot i \overset{\mathbf{(A)}}{\mapsto} c$), nor are there genuine inverses.

Less formally, as an expression has *exactly one* category, we simply use these two rules (derivation and **(A)**) to calculate the semantic value of an expression from the subexpressions. If there is no appropriate use of the application rules, then the expression has no composite meaning (it is *uninterpretable*). So we have a complete description of the formal semantics: nothing more need be said.

However, there are situations in which natural language expressions are *not* best explained in this way. Classic examples are the 'logical' parts of natural languages, the quantifiers and (natural) connectives. This ambiguity may go further[36].

We discuss this so-called *type ambiguity* in § 2. Type ambiguity occurs when a (putative) single lexeme is best described as featuring with different types in different semantic calculations. For example, **and** is best described as type $(e, (e, e))$ in NP-conjunction, but as type $((e, t), ((e, t), (e, t)))$ in intransitive VP-conjunction (§ 2.1.2).

This attempts to secure 'total typing' (where every term is typed) canonically replace simple typing with *flexible typing*, where each lexeme has *at least* one type (variously unpacked). We compare these to a proposed alternative, partial typing (§ 4), where each lexeme has *at most* one type.

Flexible typing is a broad church, only unified by the notion that occurrences of lexemes *are* differently typed in different contexts. Bar-Hillel[37] and Geach[38] allowed (syntactic) *symbols* to have several types[39]. Yet, "neither ... consider

[34] Montague (1973) claims that proper names are type $((e, t), t)$ (§ 2.5.2), yet they still have *one type only*.

[35] Lambek 1999

[36] E.g. ambitransitivity (see Kibort 2008), intensifiers, and Winter 2001 §1.

[37] Bar-Hillel 1964

[38] Geach 1970

[39] Montague's own syntax and semantics are naturally isomorphic (Liefke & Hartmann 2014).

the semantic interpretation of expressions in more than one category"[40]. The flexible typer intends to do just this. The greater *semantic flexibility* allows them to explain the troubling phenomena of § 2.

Flexible typing comes in three main strands[41].

1.8.1 Multityping

A draconian (and unpopular) account is multityping. Here, there is no mechanism for type change, nor extra words in the lexicon. The rules and lexicon are exactly as in Basic. However, lexical units simultaneously have several types. For example, **not** is type (e, e), and (t, t), and $((e, t), (e, t))$, etc.

This account will be largely excluded[42], as it suffers from some fairly decisive criticisms. Firstly, it is much more formally complex than the alternatives, with no obvious advantage. For example, multityping requires a rule to derive a new type-set from previous type-sets[43]. Such rules would radically increase the complexity of the system (and would probably yield unconvincing typesets for particular lexemes).

Further, the multityper needs some process for determining which of a lexeme's many types is used in calculating the meaning of a compound expression. But there is no obvious candidate. This suggests that multityping is structurally similar to type-shifting, but is procedurally *in*effective[44].

Finally, the semantic interpretations of multi-typed lexemes are totally unclear. In Basic, the types uniquely identified the kind of semantic value of the lexeme. It is unclear how we are supposed to interpret the meaning of a lexical entry that simultaneously has *two different* semantic kinds. For example, it is hard to imagine a single lexical entry that denotes both an entity in the domain and a predicate.

1.8.2 Type-Shifting

A more plausible account is type-shifting. Here, expressions have a single type in any context, but have "invisible semantic operators"[45]: type-shifting rules. These rules change the type of a lexeme in a semantic calculation[46]. We explore this in detail throughout § 2 and § 3.

1.8.3 Homonym Theory

Finally there is homonym theory. The homonym theorist claims that what might pre-theoretically seem to be a single lexeme in the natural language is

[40] Cresswell 1977
[41] There are marginal cases, e.g. 'argument-shifting' (Hendriks 1993).
[42] § 4.5.2 contains some discussion.
[43] Hendriks 1993
[44] See § 4.4.1.
[45] Winter 2001 § 1.3.
[46] Geach 1970, Rooth & Partee 1983

actually a collection of homonyms[47] (a 'family' with linked meanings[48]). So there are several homonyms masquerading as the single word **not**, perhaps $\mathbf{not}_{(t,t)}$, $\mathbf{not}_{(e,e)}$, etc. Just as with any other homonym, once we know *which* of the homonyms features in an expression, we can calculate appropriately.

There are two basic kinds of homonym theory. Firstly there is *stipulative* homonym theory, where each homonym is individually stipulated. There is also *rule-generated* homonym theory, where homonyms are specified by a particular rule for a given word (e.g. an and-rule for **and**). The latter is similar to type-shifting (§ 1.8.2), except that all homonyms generated by the rule are in the lexicon. The rule simply *describes* which homonyms occur. Some differences are explored in § 3.2.1.

Homonym responses are used in explanations of ambiguity outside of our main discussion[49], and are generally cohesive with other linguistic hypotheses.

[47] See H&K 1998.
[48] This 'semantic connection' has significant implication, see § 3.2.2.
[49] See Link 1998

2 Limits of Simple Typing and Flexible Typing Solutions

Many issues for the simple typer involve the *logical parts* of language. One option is to separate these lexemes from the rest of language. For example, in Categorical Grammar, quantifiers, connectives and the identity are introduced 'syncategoremically'[50], "[they] are not treated as lexical items of a particular type"[51]. Worse still, they are sometimes[52] treated as *rules* of the language, rather than proper lexemes!

However, this feels disingenuous: these words are part of natural language *just like any other*. Further, these discussions are normally limited to the connective **and** and **not**, but the same kind of problems arise for *distinctively natural* connectives like **but** (§ 2.1). Moreover, type ambiguity may *not* be limited to the logical parts of language, see § 2.5.

So we look for independent solutions. There are plausible solutions to the 'some-x-every-y' problem of quantifiers[53], so we concentrate on connectives.

2.1 Connectives

Connectives in natural languages are a systematic issue for simply typed theories.

> "The apparent simplicity of natural language coordination is one of its most enigmatic aspects."
>
> *Winter 2001 (§2)*

We are principally interested in the natural language words **and** and **or**. Negation provides a helpful introduction to the kind of problems and solutions we encounter.

2.1.1 not

The following examples of expressions are from GAMUT (1991 §4).

(6) It is not the case that Archibald cries.[54]

[50] Keenan & Faltz 1985 §I.A.1.

[51] GAMUT 1991

[52] Hendriks 1993 pp. 45

[53] Quantifiers appear in subject positions, e.g. "Everybody walks.", suggesting that they have type $((e,t),t)$. They also appear in object positions, e.g. "Everybody loves somebody." which would then have no valid calculation: $((e,t),t) \cdot (e,(e,t)) \cdot ((e,t),t)) \overset{?}{\mapsto} t$. The canonical solution is Quantifier Raising (H&K 1998). Quantifiers move to have wide scope over introduced variables, e.g. "Everybody$_1$ there is somebody$_2$ such that [t_1 loves t_2].". This consistently types quantifiers and derives the requisite multiple readings.
However, some flexibly type the quantifiers instead (Jacobson 2001).

[54] Sentential negation doesn't reduce to predicate negation. Cf. "It is not the case that Mary and John walked." and "Mary and John didn't walk"

(7) Most babies don't cry.

(8) Not every baby cries.

(9) not unkindly

Each expressions features negation acting on different kinds of subexpression, respectively sentences, intransitive verbs, (quantificational) NPs, and adverbs. So, **not** is seemingly best explained as varying by type in each expression.

In more detail, (6) is an example of sentential negation. We assumed sentences are type t, so **not** seems to have type (t, t) here. In (7), **not** is negating an intransitive verb, so has type $((e, t), (e, t))$. (8) is an example of quantifier negation. [**every baby**] is normally[55] assigned type $((e, t), t)$[56]. So here, **not** is plausibly type $(((e, t), t), ((e, t), t))$. Finally, (9) is adverb negation, which may[57] have type $(((e, t), (e, t)), ((e, t), (e, t)))$.

However, **not** seems to be a single term, a word like any other of English. So, in a simply typed theory it could only be assigned one type. Here is the inconsistency: simple typing assigns each single term a single type, **not** appears to be a single term (rather than, e.g., a family of semantically dissimilar homonyms like **rose** meaning the flower or the past tense of rise). Yet, it ostensibly has different types in different expressions.

The case-by-case analysis suggests a general pattern. Each time, **not** takes an expression of type c to an expression of type c. Generally, this suggests a (type) rule:

$$not : c \hookrightarrow c$$

Hendriks alludes to this kind of approach[58], where **not** is treated more like a *rule of the system*[59] than a lexeme in the language. His argument-shifting account seem to *split off* some connectives (principally **and**, **or**, and **not**[60]) as separate rules.

There are various reasons to be unsatisfied with this solution. For one, it is not clear the phenomenon of type ambiguity is limited to these connectives (see § 2.5) and we might hope for a more holistic solution.

Secondly, this aligns these connectives more with the abstract rules of the system, like **(A)**, than they do with the rest of the natural language. This is disingenuous: these connectives are just as much a part of language as other words. This also cuts off these connectives from their related non-rule cousins (e.g. **but**), which remain ordinary lexemes in the language.

[55] H&K 1998

[56] *Defined* common nouns have this type. However, common nouns are considered similar to predicates, with type (e, t) (sometimes justified by the semantic vacuity of the copula **is**, H&K 1998).

[57] See § 2.3.

[58] Hendriks 1993

[59] Winter 2001 §1.5

[60] Having three rules seems excessive given the inter-translatability of $\{\wedge, \neg\}$ and $\{\vee, \neg\}$

Homonym Solution

The homonym theorist suggests that though superficially **not** is a single word, it is actually a family of related words $\mathbf{not}_{x_1}$, $\mathbf{not}_{x_2}$, etc. where $\mathbf{not}_x$ is the negation operator for type x. For example there is $\mathbf{not}_{(e,t)}$, the predicate negation, which features in (7). For now we do not distinguish the stipulative and rule-generated versions of homonym theory.

Their solution to the problem is clear: each $\mathbf{not}_i$ is a different lexical entry, with a different type. So there is no *type* ambiguity at all, but more *lexical* ambiguity. So this account is *theoretically adequate*[61]: it *does* give an explanation of the phenomenon.

Significantly, the family of homonyms is *semantically linked*: there is still a common thread linking the various $\mathbf{not}_{x_i}$, some kind of abstract notion of negation. They can claim that this common thread rationalises any purported semantics for natural language which commits to a single term, **not**.

Type-Shifting Solution

There are a number of reasons to be unsatisfied with homonym theory, see § 3 (for a flavour, one might object that homonym theory is radical, as it 'introduces' too many words into the language).

Instead, one might prefer a type-shifting account. Geach presents an early version of flexible-typing[62], which employs one shifting rule:

(G) If an expression has type (a, b), then it may also have type $((c, a), (c, b))$, for any type c.

This idea is somewhat like division: if a type 'takes' expressions of type a to expressions of type b (when applied as a function), then it can also be thought of as taking expressions which are a function from c to a to expression which are functions from c to b. (In fractional notation: $\frac{a}{b} = \frac{a/c}{b/c}$.) The claim is that this rule is *totally general*, so applicable to all types of the requisite form. There is good mathematical intuition for this rule (see § 3.1).

We also require a *semantic interpretation* of **(G)**. The homonym explanation of the meaning of **not** is obvious: there are *more* lexical entries than originally supposed, but *each one* has an ordinary semantic category, like in Basic. For **(G)**, the general explanation of the conversion of the meaning, $M_{(a,b)}$, from the simpler to the more complex category is as follows[63]:

$$M_{(a,b)} \Longrightarrow \lambda y_{(c,a)} \lambda z_c [M_{(a,b)}(y_{(c,a)}(z_c))]$$

[61] See Chomsky 1965 chapter 1 §8
[62] Geach 1970
[63] GAMUT 1991 §7.3.2

Significantly, this is treated as *a single meaning*[64], which can be converted between semantic categories. It continues to constitute a single lexical entry for the lexeme.

This effectively explains some of the cases of negation. Suppose we thought that the 'base' lexical entry for negation was sentential negation, i.e. $\mathbf{not}_t$, with type (t,t). Let's suppose [**Most babies**] has type $((e,t),t)$. The (type theoretic) semantic calculation for (7) thus follows:

$$(e,(e,t))\cdot\Big((t,t)\cdot(e,t)\Big) \overset{\mathbf{(G)}}{\longmapsto} (e,(e,t))\cdot\Big(((e,t),(e,t))\cdot(e,t)\Big) \overset{\mathbf{(A)}}{\longmapsto} ((e,t),t)\cdot(e,t) \overset{\mathbf{(A)}}{\longmapsto} t$$

We concatenate the proposed types of each of the lexical units in the original sentence, preserving their original order. We then apply the newly proposed rule **(G)**, which yields a new concatenation. Two uses of **(A)** on appropriate pairs of subexpressions (perhaps determined by syntactic structure[65]) yield a single type, t, for the whole expression. This type, a truth-value, is exactly the type we (extensionally) expect for a sentence. So, this semantic calculation is descriptively adequate, it generates the right type.

This account works well for types where the starting type in the derivation chain is t (e.g. (x,t)), i.e. types whose *ancestor*[66] is t. However, it is powerless when the ancestor is e, as there is no **(G)**-application which changes the type of $\mathbf{not}_t$ from (t,t) to (e,e).

The converse is also true. On the assumption that $\mathbf{not}_e$ (type (e,e)) is the 'standard' form of negation, we cannot generate sentential negation. Generally, if the ancestor of **not** is c, **(G)** cannot change the type of **not** to a type with ancestor c', for $c \neq c'$.

This is a possible problem, as along with sentential negation ((t,t), which generates all negations with ancestor t), a language might have cases of negations with ancestor e. A possible English example is:

(eN) Mary but not John walks.

It is difficult to think that **not** is being applied to anything except **John**. So, prima facie, one might think this is an example of **not** with type (e,e).

One might attempt to explain (eN) through ellipsis, just as "Mary and John walk." is sometimes paraphrased as "Mary walks and John walks.", eliding the second **walks**.

[64] Jacobson (2003) distinguishes these as *expressions*: given an expression of type (a,c) and meanings M, there is an expression of type $((c,a),(c,b))$ with meaning M. We then calculate the compositional meaning using this distinct expression.

[65] The free use of **(A)** is somewhat controversial. Syntactic antisymmetry presumably only allows application on nodes at appropriate points in the hierarchy (Kayne 1994) . We generally apply at these nodes anyway.

[66] The ancestor is the 'highest' type, when thought of as a vertical fraction, e.g. t in $((e,t),(e,t))$ or e in $(t,((t,e),e))$. It is necessarily in the ontology.

However, such elliptical paraphrases are dubious. (VP-)ellipsis canonically targets constituents only[67]. But there is no suitable constituent of (eN) to generate a synonymous paraphrasing. Targeting **walks**, and paraphrasing as before, yields "Mary walks but not John walks.", parsed as "[Mary walks] but [[not John] walks]."[68]. But this paraphrase still requires entity negation.

If entity negation is unconvincing, there are other examples of negatable lexemes which might plausibly have types with ancestor e. For example, some adjectives (the proper noun modifiers) apparently have type (e, e), e.g. **big** in [**big John**], negated as [**not big**].

Antonyms may be more natural here. Nevertheless, *any* negation of expressions with ancestrally-e types requires a further homonym, $\mathbf{not}_e$.

The type-shifter may require at least *two* homonyms, $\mathbf{not}_e$ and $\mathbf{not}_t$. This generalises to an a-negation homonym for each type a in the ontology, as **(G)** cannot be used to shift **not** from type (d, c) to type (d, c'), if $c \neq c'$.

Even if negation-homonyms are independently questionable (entity negation is somewhat controversial), conjunction-homonyms are unavoidable for the type-shifter.

2.1.2 and

The trouble start in earnest with **and**:

(10) John walks and Mary runs.

(11) John and Mary walk.

(12) John walks and talks.

(13) Mary walks quickly and quietly.

(14) The ball is large and round.

It seems that each use of **and** is an occurrence of *the very same* lexeme. Yet **and** appears as a connective for sentences, type t, proper nouns, e, intransitive verbs, (e, t), adverbs, $((e, t), (e, t))$[69], and adjectives, (e, e)[70]. Further, this list of prototypical examples is not comprehensive. We also require conjunctions for transitive verb and quantifiers in various positions ("a few books and a lot of papers"[71]), and more[72] (see § 4.2.2).

[67] Hankamer & Sag 1976

[68] Entity negation is independently tricky. It's unclear what [**not John**] means. A putative paraphrase of [[**not John**] **walks**] is [['the opposite of John'] **walks**]. However, (1) this does not tell us if John walks, and (2) we do not have good metaphysical grasp on the entity 'the opposite of John'.

An elliptical strategy may be possible, given a descriptivist paraphrase of proper names.

[69] See § 2.3.

[70] See § 2.5.

[71] H&K 1998

[72] There are also genuinely elliptical uses of **and**: "John loves Mary, and Jack, Jill" means exactly "John loves Mary, and Jack loves Jill". GAMUT (1991) explain this contextually, but contextual dependency is not obviously necessary. *Further* homonyms are seemingly required.

The Hendriks-style rule is, for type c:

$$and : \langle c, c\rangle \hookrightarrow c$$

E.g., for intransitive verbs, **and** would have type $((e,t),((e,t),(e,t)))$.

There is no obvious explanation of this phenomenon through other (transformational) components of Basic. One putative 'explanation' of entity conjunction is an elliptical strategy, i.e. that these expressions are 'abbreviations' of longer expressions featuring a (single) sentential connective:

(15) John walks and Mary walks.

(16) Drinking and driving is unwise.

(17) *Drinking is unwise and driving is unwise.

Whilst (15) paraphrases (11), meaning is not preserved between (16) and (17): the predicate holds of the compound entity, not of each entity individually. There are equivalent arguments for compound predicates, adjectives, etc. Even worse are examples like (14): there seems no non-ad hoc way to interpret adjectival conjunction as disguised sentential conjunctions[73].

So this is a genuine problem for the simple typer. We now discuss the flexibly typed responses.

Homonym Solution

The homonym strategy carves up **and** into several distinct lexical entries, one conjunction for each type required. One option is to *stipulate* which homonyms are required. The alternative is to propose a general rule which generates the homonyms of **and**[74], e.g.:

(Conj) For *any* type c, the formal connective $\wedge_c$, of type $(c,(c,c))$, corresponds to a homonym of **and**.

This is perhaps more uniform, and alleviates the need for a reasoned argument behind each stipulated homonym.

Both versions then give similar definitions for each homonym (here, the link is *intersection*[75]). Again, the homonym theorist replaces type ambiguity with lexical ambiguity: there are simply more words.

Care is necessary, as a fully general derivation rule generates an *infinite* type system, of which infinitely many might need a conjunction, thus adding infinitely many **and**-homonyms to the lexicon[76].

[73] Translations which require positioning within an entire sentence, such as "Mary ate a large apple and that thing (which was large) was round", seem at best ad hoc and at worst non-meaning preserving.

[74] Hendriks 1993

[75] See § 4.2.

[76] The homonym theorist might stipulate a (potentially problematic) upper limit to the number of derivation applications, and so types, and so conjunctions, see § 3.2.

Type-Shifting Solution

Instead of extra homonyms, one may prefer type-shifting. However, **(G)** is insufficient, as it does not alter '3 arguments at once' in the requisite way. No combinations of the three standard rules (including composition and raising[77], see § 2.5) generate the necessary class of type-shifts either. So they require a new **(G)**-style rule:

$$\textbf{(G2)} \quad (x,(y,z)) \mapsto ((c,x),((c,y),(c,z)))$$

where x, y, z, and c are types. Fractionally:

$$\frac{\frac{x}{y}}{z} \mapsto \frac{\frac{x/c}{y/c}}{z/c}$$

(G) and **(G2)** suggest a general pattern of G-rules, for any number of layers in a fractional hierarchy[78]:

$$\textbf{(Gn)} \quad (x_1,(x_2,...(x_{n-1},x_n))) \mapsto ((c,x_1),((c,x_2,),...((c,x_{n-1}),(c,x_n))))$$

For **(G2)**, if a term has type $(x,(x,x))$, then at any point in a calculation it can shift to type $((x,y),((x,y),(x,y)))$. Assuming that the 'basic' meaning of **and** is sentential, i.e. $(t,(t,t))$, we can analyse (12) like so:

$$e\cdot(e,t)\cdot(t,(t,t))\cdot(e,t) \overset{\textbf{(G2)}}{\longmapsto} e\cdot\Big((e,t)\cdot((e,t),((e,t),(e,t)))\cdot(e,t)\Big) \mapsto e\cdot(e,t) \mapsto t$$

The type-shifter still has an issue with shifting between types with ancestor e and ancestor t. We have unambiguous examples of sentential and entity conjunction. The respective expected types are $(t,(t,t))$ and $(e,(e,e))$. But there is no combination of the standard type-shifting rules (§ 2.5) which shift between these two. So the type-shifter is committed to at least two homonyms: $\textbf{and}_e$ and $\textbf{and}_t$.

2.1.3 or

There do not seem to be any unique, distinctive phenomena for **or**, so analysis proceeds parallel to with **and**. Exactly similar solutions can be proposed by the flexibly typed accounts, e.g. **(G2)** with two homonyms $\textbf{or}_e$ and $\textbf{or}_t$ for the type-shifter.

[77] Humberstone 2005

[78] See § 3.1. **(G2)** *is not* mathematically motivated, as it is not a theorem of the natural algebra. **(Gn)** is a theorem iff n is odd.

2.1.4 but and because

The case of distinctively *natural* connectives is interesting.

For example, **because** typically connects sentences, e.g. "I eat on the move because I am busy". It can also be used as a connective between a sentence and a noun: "I eat on the move because of business". However, the total typer can resolve this by saying that [**because of**] has a separate lexical entry to the term **because** (a homonym-style solution).

Some, like [**as well as**], look all but synonymous with other connectives, in this case **and**.

More interesting is **but**, which has a distinctive (ordered) contrastive effect.

(18) She is poor but honest.

(19) Mary walks but John runs.

(20) The philosophers, but not all the academics, went skiing.

The problem is just as before, and the solutions are the same again.

We might note that the solution can be an extension of the solution for **and**. Suppose the definition of [a **and** b] is $[\![a \text{ and } b]\!]$. The flexible typer could then define **but** as $[\![a \textit{ but } b]\!] = [\![a \text{ and } b]\!] \wedge [\![a \text{ constrasts with } b]\!]$[79]. Similar analyses can be given for, e.g. **yet**.

2.1.5 if

but is an example of a 'weakly' asymmetric binary connective, whilst **if** is a 'strongly' asymmetric connective.

(21) I'll sleep if I'm tired.

(22) He is reliable, if irritating.

(23) a significant, if small change

(24) He spoke honestly, if hurtfully.

(25) She drank water, if at all.

As ever, in these expressions **if** is seemingly best explained as varying by type.

However, the ambiguity here is different. Most plausibly, these are examples of *distinct lexical entries* for the same syntactic string. A credible analysis for (25) is as an idiomatic elliptical expression (roughly "If she drank anything, it was water").

(22) and (23) feature **if** working comparatively rather than conditionally, for instance (22) ostensibly means "He is reliable although he is irritating.".

[79] The expression [**contrasts with**] causes its own typing problem, so the homonym account may be more effective here.

Neither case is obviously connected to conditionalisation. We can see this by attempting to analyse (22) as sentential ellipsis, as in § 2.1.2. The go-to elliptical analysis would be:

(22') He is reliable if he is irritating.

(22) is clearly not synonymous with (22'). Indeed, replacing **if** with **but** works better. Something similar can be said about (23) and (24).

These are more plausibly different lexical entries for the same term, i.e. semantically *dissimilar* homonyms.

2.2 Logical = Type Ambiguous?

So far only *logical* parts of language (connectives, quantifiers[80]) have suffered from type ambiguity. Characterising 'logical' is difficult. Attempts typically say things like 'those lexemes whose lexical entry requires only formal logic'. This might be unsatisfactory, as it's somewhat uninformative and appears vulnerable to circularity.

§ 2.1 might motivate the hypothesis that the 'logical' parts of language are precisely the expressions that suffer from type ambiguity. This would include words which aren't considered *strictly* logical, e.g. **but**, but this isn't a serious concession, e.g. it's plausible that we could define **but** without going far beyond formal logic[81].

Justifying this hypothesis would be a major theoretical boon, as it would solve the independent problem in the philosophy of logic and language.

However, there are words which seem not to fit this pattern.

2.3 Adverbs

The distinctively *non*-logical adverbs and adjectives may also suffer from type ambiguity. These problems are general: rather than *particular* adjectives or adverbs being ambiguous, the whole class is. For example, the problem for adverbs is that an apparently single adverb modifies differently typed verbs (e.g. transitive and intransitive verbs).

We look at adverbs first, where accounts bifurcate. Firstly, there are *extensional* approaches, like Basic, which has a restricted ontology: just e and t. However, this is not standard. The alternative is *event semantics* (§ 2.4).

Suppose we continue extensionally[82]. Consider the following examples:

(26) John walked aggressively.

[80] See footnote 53.
[81] Though the natural connectives are sometimes (ambiguously) non-truth-functional.
[82] Standard objections are explored in § 2.4.

(27) Fido aggressively bites John.

(28) Mary aggressively gave John the pen.

The plausible types for **aggressively** are:

(26) $((e,t),(e,t))$,

(27) $((e,(e,t)),(e,(e,t)))$,

(28) $((e,(e,(e,t)),(e,(e,(e,t)))))$.

Nonetheless, they seem to be *the very same word*. The situation is apparently parallel to with connectives (§ 2.1). Again, the types form a regular pattern.

One quick response is to claim that adverbs modify *VPs* (verb phrases), not verbs. The above assumes that the adverb modifies the *verb alone*. E.g. **aggressively** modifies **gave** (to form [**aggressively gave**]), rather than modifying [**gave John the pen**]. One might reject this assumption, saying that the adverb modifies the VP, e.g. [**aggressively** [**gave John the pen**]]. So, **aggressively** could have just one type: $((e,t),(e,t))$.

However, (di-)transitive *compound verbs*[83] suggest otherwise.

(29) Fido aggressively bit and then lovingly licked the postman.

(30) John slowly read and then generously gave Mary the book.[84]

Elliptical strategies seem not to preserve sentence meaning, and also seriously contravene the intuitive structure of the sentence. So we seem forced to say adverbs modify verbs not VPs.

This suggests that (26)-(28) are instances of adverbs varying by type in each expression.

As ever, there are possible flexible typing solutions. The homonym solution is clear. Type-shifters take **aggressively** to 'normally' have type $((e,t),(e,t))$, then apply **(G)** to yield the requisite types.

$$((e,t),(e,t)) \overset{\mathbf{(G)}}{\longmapsto} ((e,(e,t)),(e,(e,t))$$

$$((e,t),(e,t)) \overset{\mathbf{(G)}}{\longmapsto} ((e,(e,t)),(e,(e,t)) \overset{\mathbf{(G)}}{\longmapsto} ((e,(e,(e,t)),(e,(e,(e,t))))))$$

[83] Informal polls of competent speakers suggest that these compound verbs are legitimate, grammatical expressions of English.

[84] Concluding ditransitive verbs with heavier NPs make expressions appear more grammatical e.g. [**the book she wanted to read while she was ill**] (cf. proper names). We are cautious here, as heavy NPs could mask other instances of marginal grammaticality. Short definite NPs are a compromise.

2.4 Diversion: Event Semantics

The 'extensional' (e-t) account suffers from independent problems. We outline some, then show that the aforementioned type ambiguity 'carries over' to a simply typed account with an ontology enlarged by an extra type, s, the event. This justifies our simplification to Basic.

We have so far assumed that some plausible analysis of adverbs in Basic is possible, i.e. we can define and assign a plausible type to each adverb. We reasoned that *particular* candidate types are implausible, as *single* adjectives modify verbs of different types.

However, some arguments suggest that *any* semantic analysis in Basic will be implausible, independent of type ambiguity, as the candidate definitions are inadequate. The classic literature is heavy with examples, suggesting that we need richer semantics. We provide a sketch:

(WZ1) John danced beautifully[85].

John and **danced** have their standard definitions, *john* and $\lambda x.\textsc{danced}(x)$. There is no *event* variable, so using the extensional expression $\textsc{beautiful}(x)$ would generate the wrong meaning, that *John* is beautiful, not the dancing-event. Instead, we assume that adjectives are verbal modifiers of type $((e,t),(e,t))$. We are forced to define **beautifully** as something like:

$$[\![beautifully]\!] = \lambda f_{(e,t)}.(f_{(e,t)}\ beautiful)$$

But this definition doesn't make sense. $[\![beautifully]\!]$ doesn't characterise a legitimate function, it's just an uninformative concatenation of English words. It does not map f to anything in the metalanguage. In particular, the interpretation of '*beautiful*', the string which features in $[\![beautifully]\!]$, is unclear. The string $\lambda e.(\textsc{danced}(e)\ beautifully)$ is simply not a meaningful VP! So semantic difficulties cast doubt on $((e,t),(e,t))$ as a type for adverbs.

The alternative e-t analysis of adverbs introduces a new (e,t)-predicate, $\textsc{bd}(\textsc{x})$, as the denotation of [**danced beautifully**], then formalise (WZ1) as:

(WZ2) $\textsc{bd}(john)$

However, intuitively we can logically infer from (WZ1) to:

(C) John danced.

Yet the e-t formalisation does not obviously legitimate the inference from (WZ2) to (C).

[85] Winter & Zwarts 2011

Instead, some contemporary accounts of compositional semantics enlarge their ontology, adding in a new basic type, s, the event (or situation). This *event semantics* tradition started with Davidson[86], and now effectively explains various linguistic phenomena. Crucially, it more plausibly explains adverbial modification. For example, (WZ1) can be formalised as:

(WZ3) $\exists e$ DANCE$(e, john) \wedge$ BEAUTIFUL(e).

where BEAUTIFUL(e) is an ordinary (s,t)-function. This generates the required meaning for the sentence, that the *event* is beautiful, rather than John. It clearly legitimates the inference from (WZ1) to (C), as (C) 'is' the first conjunct of (WZ3).

The extensional semanticist also struggles to formalise sentences featuring other verbal modifiers, e.g. locative, temporal and instrumental modifiers[87], whilst event semantics provides convincing analyses.

(D1) Jones buttered the toast with a knife in the bathroom at midnight.

Extensional semanticists could assert that **buttered** has several silent variables, so is properly formalised by a five-place relation BUTTERED$_5$*(x,y,m,l,t)*. But this is untenable. Firstly, there is little intuitive evidence for the silent variables in the unmodified "John buttered the toast.". More definitively, there does not seem to be a *fixed number* of variables (the number of modifiers does not seem limited).

Instead, they could stipulate a new composite transitive VP, BUTTERED$_2(x, y)$, the meaning of [x **buttered** y **with a knife in the bathroom at midnight**]. But this introduces massive semantic ambiguity. Different combinations of modifiers force **buttered** to have a different meaning (e.g. BUTTERED$_{loc}(x, y, l)$ for [x **buttered** y **in** l]). But we think that **buttered** has *the same* meaning irrespective of modifiers, which only provide extra information about the buttering. This ambiguity also leads to problems with natural implications, as with (WZ1) and (C).

Meanwhile, the event semanticist can formalise (D1) naturally, whilst avoiding the above issues:

(D2) $\exists e$ BUTTERED(*jones, the toast, e*) $\wedge$ WITH(*e, a knife*) $\wedge$ IN(*e, the bathroom*) $\wedge$ AT(*e, midnight*).

Finally, event semantics adequately explains the putative type ambiguity of adverbs. **aggressively** has one definition $\lambda f_{(s,t)} \lambda e.[f(e) = 1 \wedge$ AGGRESSIVE$(e)]$, and so a unique type (s,t). The denotations are otherwise the same, except for verbs, which are *also* analysed as type (s,t), e.g. $[\![walked]\!] = \lambda e.$WALKED$(e)$. A

[86] Davidson 1967
[87] Laseroshn 2006

special head, (little) v, adds the subject above the VP. For simplicity, we use theta role heads[88], e.g. $\lambda e \lambda x.[\text{AGENT}](e, x)$, for "$x$ is the agent of e".

(26) $\exists e$ WALKED$(e) \wedge$ AGGRESIVE$(e) \wedge [\text{AGENT}](e, john)$

(27) $\exists e$ BITES$(e) \wedge$ AGGRESIVE$(e) \wedge [\text{AGENT}](e, fido) \wedge [\text{THEME}](e, john)$

(28) $\exists e$ GAVE$(e) \wedge$ AGGRESIVE$(e) \wedge [\text{AGENT}](e, mary) \wedge [\text{THEME}](e, the\ pen) \wedge [\text{GOAL}](e, john)$

This permits a unique definition for each adverbial connective. For example, adverbial-**and** takes two (s,t)-variables to their intersection[89]:

$$[\![and]\!] = \lambda f_{(s,t)} \lambda g_{(s,t)} \lambda e.e \in \{e' \mid g(e') = 1 \wedge f(e') = 1\}$$

E.g., in (13):

$$[\![quickly\ and\ quietly]\!] = \lambda f \lambda e.e \in \{e' \mid \text{QUICK}(e') = 1 \wedge \text{QUIET}(e') = 1\}$$

2.4.1 Problems Carry Over

The event semanticist successfully explains adverbs (assigning them type (s,t)). However, they have no extra resources for explaining the type ambiguity of adjectives in § 2.5.

Moreover, the connectives are still type ambiguous in simply typed event semantics. For example, **and** can conjoin expressions of varying types. Recall the example:

(14) The ball is large and round.

The expected meaning of **and** here is:

$$[\![and]\!] = \lambda f_{(e,t)} \lambda g_{(e,t)} \lambda x.x \in \{x' \mid g(x') = 1 \wedge f(x') = t\}$$

i.e. taking two variables of type (e,t) to their intersection, *not* two of type (s,t) as above. Basic and simply typed event semantics suffer from the same type ambiguity for connectives (§ 2.1).

Of course, there are type-shifting and homonym analogues for event semantics, which provide analogue solutions to the type ambiguity in event semantics in the obvious way.

This diversion yields two conclusions. Firstly, there are certain linguistic phenomena[90], particular adverbs, which the simplified extensional accounts are too weak to explain effectively. We can solve this by adding a third item, the event, to the original ontology. This is called event semantics.

[88] Winter & Zwarts 2011
[89] Champollion 2014(a)
[90] Davidson 1967.

Secondly, even in this more sophisticated semantics, simple typing causes type ambiguity, at least amongst the connectives. Aside from the case of adverbs, the event semanticist has no extra solutions to type ambiguity problems.

This ends the diversion. We now mainly stay in the extensional realm of *e*-*t* analysis, à la Basic.

2.5 Adjectives

Unlike adverbs, contemporary accounts of adjectives do not essentially employ the extra type s. The analysis of nouns given by the event semanticist is the same as in Basic. So we consider Basic.

(31) big John

(32) big dog[91]

The adjective seemingly modifies the NP subexpression each time, so the whole composite expression should have the same type as the subexpression[92]. So, [**big John**] has type e (like **John**), and [**big dog**] should have type (e,t). Hence, the types of **big** are (e,e) and $((e,t),(e,t))$ respectively.

Neither **(G)** nor any generalisations **(Gn)** appropriately shift **big**. Rather than stipulating a further rule to cover this case (alone!), the canonical approach reanalyses the type of the *proper name*, not the adjective.

Here, there is another bifurcation. There are those who broadly agree with sketch the above, who propose a further type-shifting rule for proper names, and those who advocate a different *base type* for proper names.

2.5.1 Raising and Composition

Those who accept that names are type e explain the type ambiguity by introducing two further rules, raising and composition[93]:

$$\textbf{(M)} \qquad a \mapsto ((a,b),b)$$

$$\textbf{(C)} \qquad (a,b)\cdot(b,c) \mapsto (a,c)$$

where a, b, and c are categories. The equivalent semantic interpretations are fairly clear, **(M)** corresponds to the following conversion of the expression's meaning, M_a, from the simpler to the more complex category[94]:

$$M_a \Longrightarrow \lambda y_{(a,b)}[y_{(a,b)}(M_a)]$$

[91] Adjectives seem to modify *undefined*, rather than defined, NPs (H&K 1998 §4.3).

[92] Saliency conditions may be required (H&K 1998 §4.3.3).

[93] Humberstone 2005

[94] For type e expressions, **(M)**-shifting correspond to *generalised quantifier meanings* (Jacobson 2001).

Whilst **(C)** corresponds to ordinary functional composition[95]. So, if the (a, b)-type expression has meaning $\lambda y_a.M_b$ and the type-(b, c) expression has meaning $\lambda y_b.M_c$, then the composite expression has meaning $\lambda y_a.[M_c(M_b)]$.

In fractional notation (e.g. **(M)** is $a = \frac{b}{b/a}$), both are theorems of the natural algebra on the types[96]. The solution here is rather cunning: **big** *doesn't* change type, the *name* does instead. So **big** is unambiguously type $((e, t), (e, t))$, respecting (31).

The semantic analysis for the composite expression follows as expected:

$$e \cdot ((e,t),(e,t)) \overset{\mathbf{(M)}}{\longmapsto} ((e,t),t) \cdot ((e,t),(e,t)) \overset{\mathbf{(C)}}{\longmapsto} ((e,t),t)$$

So, the modified proper name has type $((e, t), t)$, like a defined NP[97].

2.5.2 All Nouns are General

Instead, Montague[98] rejects the claim that **John** is best analysed as type e. He argues that *all* NPs must have the same type. In a word, Montague takes the intuitive category 'noun' seriously. As quantified NPs have type $((e, t), t)$, so too proper names must have type $((e, t), t)$.

This also provides a uniform solution to the problem of conjunctions of proper names and quantifiers[99], e.g. "Mary and every other girl". Assuming that conjunction is type-symmetric (it only conjoins like types), and the implausibility of the expression **[every other girl]** denoting an individual (type e), **Mary** is forced to have type $((e, t), t)$[100].

I interject a small comment. Montague's account feels strange, and possibly even counter-intuitive. Firstly, we *can* plausibly define **and** to include such asymmetric conjunctions (see footnote 158). Secondly, and more controversially, if *any* class of natural language expressions refer to *particular objects in the domain* (i.e. entities), it must include the proper names. It's not clear why we should value Montague's everyday *categorisations* (all NPs have the same type) over these everyday meanings for each term[101].

[95] Dowty 1988

[96] Basic has no '0' type, which is otherwise problematic (§ 3.1).

[97] Applying 'backwards' **(M)** then yields e for **[big John]**, but is controversial (Partee 1987).

[98] Montague 1973

[99] Barwise & Cooper 1981

[100] This 'generalised quantifier' means $\{S \subseteq D = D_{e,D} : mary \in S\}$.

[101] Fine-grained semantic intuitions are questionable, see § 3.3.

2.6 A Cautionary Example: The Lexical Ambiguity of 'that'

Ambiguity extends much further than type ambiguity. Phonological ambiguity, such as homophones (**rode**, **road**), are unproblematically distinguished. Certain homonyms are also clearly distinguishable (**bank** to store money, **bank** of a river). The latter is generally called *semantic* ambiguity, but is in a sense phonological: these two semantic units are *said* the same way (but they might easily not have been).

Moving further from this 'concrete' fragment of language (which includes some nouns, verbs, etc.), distinguishing kinds of ambiguity becomes increasingly difficult. A cautionary example is **that**.

(33) She answered that she was not hungry.

(34) The mountain that has the yellow grass is tall.

(35) He's not that fat.

In (33), **that** introduces a subordinate clause. In (34), it is a complementizer, which introduces a relative clause. If forced to assign these uses types, we might suggest (t, t) and $(e, ((e, t), t))$ respectively (but they are more plausibly *semantically vacuous*[102]). They are somehow more *abstract* than other parts of language, like **mountain** or **yellow**[103]. However, in (35), **that** is being used totally differently. It is a modifier, perhaps type $((e, t), (e, t))$.

Again, we have a word which appears to have different types in different expressions.

The first indication that this is different to connectives or modifiers is that the candidate types do not fall into a neat, obvious pattern as we had before.

So *what kind* of ambiguity is this really? The adverbial use is clearly distinguishable from the other two. We can easily imagine **that** not having this meaning. In (35), **that** could be replaced by **overly** whilst preserving meaning (unlike in the other examples). Adverbial-**that** is clearly semantically unrelated to the other occurrences.

Untangling the uses in (33) and (34) is more difficult. A potential explanation for this is *abstractness*. Here, **that** is not so grounded in the world (it doesn't pick out an entity or a property), so any putative semantic intuitions are weakened.

Rather than type ambiguity, this is more plausibly *lexical* ambiguity: **that** has different lexical entries with different types (or certain instances are semantically vacuous). There is no good reason to think that **that** has *a single* lexical entry which must be type-shifted to explain its varying occurrences.

[102] H & K 1998 §5. Some occurrences might be meaningless sub-strings of larger atomic expressions, e.g. [**believes that**].

[103] There is also demonstrative-**that**, e.g. "That mountain there is yellow.". Davidson (1968) argues that demonstrative-**that** and relative clause-**that** are the same lexeme, but non-English language data suggests otherwise.

2.7 Which of These are Important?

We can broadly characterise the above problems for the simple typer into two categories:

1. Connectives (particularly **and**, **or**, and **not**),
2. Modifiers (maybe just adjectives).

Some non-logical language problems have plausible piecemeal solutions. For adverbs, type-shifting within event semantics is satisfactory. For adjectives, there is Montague's solution (§ 2.5.2), or type-shifting using **(C)** and **(M)**.

So, it seems that the *central* issue for the simple typer is explaining the connectives[104]. Hence, we focus on how effectively the flexible typer (and the mixed account, Mix) can explain the problem cases presented by the connectives, along with the theoretical motivations for their accounts.

[104]Further phenomena are worth investigating, for example non-event explanations of ambitransitive verbs (Kibort 2008), and intensifiers, like **very**.

3 Limits of Flexible Typing

Simply typing is not descriptively adequate (the basic requirement for a semantic theory). Homonym theory and type-shifting correct for this.

We sketch some *particular* criticisms facing the flexible typers. We thereby cast doubt on the whole enterprise of 'total typing'. The main issue is theoretical parsimony, though we begin with the problem of mathematical motivation.

We have discussed three rule-patterns: **(M)**, **(C)** and **(Gn)** for various n. These raise two linked questions:

1. What motivates these rules?
2. How does the type-shifter justify their *semantic* effects?

Just as the philosophically-minded physicist aims for *fundamental, universal* rules in physics[105], the semanticist hopes to find the fundamental, universal rules of semantics. However, so far the only justifications for the rules have been: (1) mathematical naturalness, and (2) theoretical adequacy. Let's consider these in order.

3.1 Mathematical Motivation

When (mathematically) modelling any natural system, we use some data to justify some candidate general rules governing the system. We then investigate the effects of general rules of the mathematical model, and then test these extrapolations against further evidence from the system. We hence evaluate the fit of the mathematical model.

In our case, there is a natural algebra on Basic, a kind of pre-group, which is suitably closed under derivation and **(A)**. The principal type-shifting rules are **(G)**, **(M)**, and **(C)**. Each (provably) represent a theorem of this natural algebra.

In fractional notation the base rules are:

$$\textbf{(G)}\ \tfrac{a}{b} \mapsto \tfrac{a/c}{b/c},\ \textbf{(M)}\ a \mapsto \tfrac{c}{c/a},\ \text{and}\ \textbf{(C)}\ \tfrac{a}{b} \cdot \tfrac{b}{c} \mapsto \tfrac{a}{c}$$

There is no worry about $c = 0$ cases for **(M)**, which cause problems in other division rings, as the type-shifter assumes that there is no '0' type in the basic ontology (and total typing), so **(M)** is applicable to the whole algebra.

This mathematical motivation is a serious boon. These type-shifting rules are in some sense *expected* of the natural algebra, so the type-shifter can reasonably say that the natural-algebra *is* a good model for their semantics.

[105] Frigg & Hartmann 2012

However, in other ways, the type-shifter's solution diverges from this model. So we might think that the model is unsuitable for type-shifting. Meanwhile, the alternative models are radically unsuitable for even the simplest expressions. These together go some way in discrediting type-shifting as an adequate account of natural language semantics.

The divergence comes in three strands:

1. The type-shifter does not commit to other (sufficiently simple) theorems of the algebra, some of which would generate unsuitable meanings.
2. The type-shifter does not assert the rules in the full generality or equality of the theorems.
3. Some type-shifting rules are not theorems of the algebra. Worse still, the alternative algebras which account for these rules yield puzzling side-effects.

Before unpacking these, we dismiss some possible responses.

One putative way to avoid such criticism is a wholesale rejection of the importance of mathematical motivation. Perhaps such motivation is good supporting evidence for the rules which *are* so motivated, but it does not undermine the other rules, which are independently motivated by their necessity in explaining the data.

For example, to avoid deviant type-shifts, the type-shifter might limit their rules (by sublexicon or type-form, § 3.2.3). However, such limitations are not mathematically justified. This might strengthen the claim that mathematical motivation is *un*necessary, it only muddies the water. Instead, the type-shifter searches for explanatorily adequate rules of semantics, irrespective of the mathematical model they generate.

This kind of response seems unsatisfactory for me. Firstly, the singular emphasis on explanatory adequacy is questionable. It is not clear that the type-shifting rules *are* necessary. Indeed, homonym theory (and Mix, § 4) claim to be explanatorily adequate, without using such rules.

Moreover, the outright rejection of the project of mathematically modelling compositional semantics is troubling. Internally to semantics, having no mathematical model limits the kind of reasoning we can do about the system[106], for example, it is seemingly more difficult to prove that the process of ordinary semantic calculations is decidable.

Externally to semantics, and more abstractly, we expect natural systems to be adequately describable in mathematical terms[107] (to some admissible approximation, which should include purportedly fundamental type-shifting rules!).

[106] Buszkowski 1997
[107] Wigner 1960

Natural languages are apparently typical amongst natural systems. So it is suspect to say that natural languages *contravene* axiomatic naturalism. It seems more reasonable to suggest that if the natural algebra is unsuitable, this is not good evidence for the 'un-modelability' of natural language semantics.

3.1.1 Other Theorems

What status do the *other* theorems of the natural algebra have?

(M), **(C)**, and **(G)** are theorems, but there are plenty of other theorems which type-shifters *do not* use, e.g. **(G2n+1)** for large n.

Firstly, it seems arbitrary to say that the theorem-hood of these rules motivates their position amongst the fundamental rules of compositional semantics, but other theorem-hoods do not motivate other rules. Even if we limit rules by type complexity[108], there are numerous theorems which are not claimed by the type-shifter, for example:

$$\textbf{(C2)} \quad \tfrac{x}{a} \cdot \tfrac{a/b}{c} \mapsto \tfrac{x/b}{c}$$

which corresponds to $(a, x) \cdot (c, (b, a)) \mapsto (c, (b, x))$, along with higher order theorems, e.g. Lambek's rule[109]:

$$\textbf{(CP)} \quad \Gamma b \mapsto a \Rightarrow \Gamma \mapsto \tfrac{a}{b}$$

A limited rebuttal is to accept further rules, for example internal composition: $((a, x), (x, c)) \mapsto (a, c)$. This could be useful in *reducing* types when several type-shifts have been applied. But such a response is unsatisfactory, it does not explain why *some* theorems are taken as rules, and others are not.

3.1.2 Limitations on Generality and Equality

If the type-shifter motivates the type-shifting rules by their theorem-hood, we would expect them to have theorem-like properties. The relevant theorems are *totally general*, they apply to expressions of any type, and they are equalities, so apply in 'both directions'.

However, the type-shifter seems forced to limit the rules on both fronts. As presented, **(M)** and **(G)** only increase complexity, e.g. **(G)** *raises* type (c, d) to type $((x, c), (x, d)))$. They are one-directional processes. There is no obvious mathematical reason why reductions of complexity using these equalities would be illegitimate.

Meanwhile, these converse applications would generate issues for the type-shifter. For example, 'backwards' **(G)** can type-shift from $((e, t), (e, t))$ to (t, t).

[108] Suitable measures of complexity include 'number of derivation rules used to derive the category' or the syntactic measure 'number of commas'.

[109] Humberstone 2005.

Consider the intensifier **very**. It can be used as an ad-adjective[110], i.e. has a type $((e,t),(e,t))$ meaning. By 'backwards' **(G)**, it has a meaning of type (t,t), i.e. a sentential modifier. But there is no such use of **very**!

Moreover, this problem is general: there are *no* clearly legitimate uses of 'backwards' **(G)**, or the other rules[111]. Nevertheless, we are owed a principled argument for why the type-shifting rules are one-directional, beyond descriptive adequacy.

The mathematical model of their system motivates rules far beyond what the type-shifter uses, yet they have not given us strong reason to accept only their rules.

3.1.3 Alternative Algebras

To explain the type ambiguity of *binary* connectives, a further G-style rule was suggested:

$$\textbf{(G2)}\ \tfrac{x/y}{z} \mapsto \tfrac{\frac{x/c}{y/c}}{z/c}$$

This is *false* in the same natural algebra, it is *not* a theorem. There are 'too many' occurrences of c on the right hand side of this equation[112]. Indeed, **(Gn)** is true in the natural algebra iff n is odd.

One response by the type-shifter is to question the suitability of the 'natural algebra' on Basic as a model for semantics. The outright rejection of *any* mathematical model is unsatisfactory (§ 3.1). The only alternative is to say that a *different* model is suitable.

Without going into full details, the structure of Basic seemingly requires the model to be some kind of type-*algebra* (loosely defined). One might rationalise an *alternative* algebra by claiming that the initial presentation of Basic is misleading. The fractional structure is *not* like ordinary division. Instead it is better to think of **(G2)** as saying roughly: "(x 'requiring' y) requiring a z maps to an x requiring a c requiring..." etc.

Such an algebra on the terms *could* be stipulated. It needs a variant application rule, where a type-c term is applied to a type-$(x_1/y_1,(x_2/y_2,...,x_n/y_n)...)$ term at 'all levels' of the fraction at once. More formally, we replace **(A)** with this 'hierarchical application':

(*A) For any types c_1,...,c_n,d: $d \cdot (...(d,c_1),...,(d,c_n)...) \mapsto (...(c_1,...,c_n)...)$

[110] Keenan & Faltz 1985

[111] "[backwards **(M)**] is not necessarily part of the grammar of English at all", Partee 1987

[112] The right hand side is the left hand sided 'divided by c'.

Each **(Gn)** *is* a theorem of this new algebra.

However, this algebra is *not* suitable as a model of compositional semantics for natural language. Specifically, **(*A)** generates *incorrect* meanings for expressions almost immediately. For example, **(*A)** validates derivations like the following:

$$(e,(e,t))\cdot e \overset{\textbf{(*A)}}{\longmapsto} t$$

i.e. from a proper name and a transitive verb, we calculate a truth value, for example the generated meaning of [**loves Mary**] would be "*Mary* loves Mary.". All kinds of intuitively way-off semantic analyses are legitimated. The opposite is also true: "John loves Mary" would not have a composite meaning:

$$e\cdot(e,(e,t))\cdot e \overset{\textbf{(*A)}}{\longmapsto} e\cdot t \longmapsto ?$$

This formal system is descriptively inadequate.

Further, it introduces calculation ambiguity where the ordinary system is procedurally effective. For example, in (26), we can validly calculate as required:

$$e\cdot((e,t),(e,t))\cdot(e,t) \overset{\textbf{(*A)}}{\longmapsto} e\cdot(e,t) \overset{\textbf{(*A)}}{\longmapsto} t$$

However, we could *also* calculate like so:

$$e\cdot((e,t),(e,t))\cdot(e,t) \overset{\textbf{(*A)}}{\longmapsto} (t,t)\cdot(e,t) \overset{\textbf{(C)}}{\longmapsto} (e,t)$$

The calculation algorithm in the new algebra would need some way to prioritise the first calculation[113].

So here is the zugzwang:

1. If the type algebra is natural, then **(G2)** is not a theorem, so we cannot motivate the appropriate type-shift for binary connectives.
2. If the algebra is *un*natural, then application works entirely differently, and the whole edifice of type theoretic formal semantics is unrecognisable.

The only option available to the type-shifter seems to be taking these additional rules as something like *axioms* (perhaps this corresponds to some kind of 'quotient structure' on the type hierarchy). However, further worries about the theoretical strength of such an axiom may put these rules in doubt (§ 3.2.3).

[113] A rich syntactic theory might suffice, by limiting **(*A)** to nodes in the syntactic tree.

3.2 Parsimony

Another class of objection concern *parsimony*. Flexibly totally typed systems are *not* parsimonious. In general, they have far too much conceptual strength.

3.2.1 Homonym Theory and Parsimony

Both the stipulative and rule-generated homonym theorists are committed to a large number of *additional* lexical entries[114] in the natural language. This approach appears more complex and *more lexically ambiguous* than non-homonym explanations, as homonyms are more prevalent. Further, the rule-based homonym theorist seems committed to an implausible lexicon, whilst it is not clear that the stipulative homonym theorist *even knows* which homonyms they are committed to. This is damaging in several ways.

A. Firstly, (each form of) homonym theory seems to be *radical*. They are forced to say that the lexicon is somehow longer than 'expected'. We characterise the homonym theorist as 'having more homonyms', or that they 'stipulate additional meanings'. This suggests that there was a initial (possibly pretheoretic) number of homonyms with which we theorise. The homonym theorist must reject any *initial* number of homonyms, irrespective of the evidence for it[115]. In this sense, homonym theory is *radical*, rather than *therapeutic*.

This radicalism may not be so serious. They might respond that as long as their account is explanatory and otherwise defensible, then their revision of the initial theory is unproblematic. After all, the *length of a lexicon* (or the number of entries for each syntactic string) is a decidedly abstract notion. Given that the initial theory could even be pretheoretic, a competent speaker might (unreflectively) endorse some number of homonyms for a word, but might adjust this commitment on greater reflection (or semantic training).

One quick counter-reply here is that whilst the *length of the lexicon* may be abstract, closely connected notions like 'the size of the language' or 'the number of words' are more concrete ("it seems quite probable that English has more words than most comparable world languages."[116]). How exactly these notions are connected is unclear, but languages, as natural systems, must surely be measurable, and the homonym language seems somehow 'larger'.

B. Two specific issues for *stipulative* homonym theory are specifying exactly which homonyms they endorse, and giving an adequate account of language acquisition. For example, § 2.1.2 contains a non-exhaustive list of five cases where **and** is assigned pairwise distinct types. The homonym theorist must give an account of *which* homonyms of **and** they endorse, and why these ones exactly.

[114] Or *additional words*, which seems worse.

[115] Such evidence might come from (controversial) competent speaker intuitions that, for example, **and** has one (main) meaning.

[116] OED Blogs 2016.

The stipulative homonym theorist also seems to require a competent speaker to have some knowledge of each distinct homonym in order to learn a language. These homonyms are possibly numerous and obscure, e.g. ditransitive verb-conjunction, $\mathbf{and}_{(e,(e,(e,t)))}$. However, (simple) theories of language learning suggest we *do not* need to be acquainted with each homonym (particularly those with obscure types) to understand the apparent single lexeme, or to be a generally competent speaker[117].

C. Rule-generated homonym theory suffers from a host of problems.

Firstly, a homonym-generating rule not only inflates the lexicon, but also increases the *complexity* of the system. The additional homonyms are a function of some structural properties of the system (homonyms are generated for *every* type), rather than a function of the properties of the word. This seems to detract from the formal simplicity which justified homonym theory.

It is also similar to type-shifting without its advantages. Unlike type-shifting, there is greater semantic ambiguity as all of the homonyms *are realised* in the lexicon (whilst a type-shift can be viewed as a silent operator in the expression, adding no further lexical ambiguity).

An *unlimited* homonym-generation rule is independently problematic. We have discussed why the derivation rule might be limited (§ 1.6.2). If it is *unlimited*, and a homonym-generating rule generates a homonym for *every* type, then it yields a deeply puzzling result: an infinite lexicon and 'infinite lexical entries'.

We normally think that natural language lexicons are finite[118]; certainly our dictionaries are finite. Presumably we cannot hold an infinite number of discrete, *realised* lexical entries in our heads. In general, it is not clear how any naturally occurring system could have infinitely many components[119]. However, if the type system is infinite, and if just one homonym-generation rule is *un*limited, then their lexicon is infinite.

Rule-generated homonym theory also suffers from language acquisition problems. If the rules are *un*limited, the language may have an infinite lexicon. It's unclear how a language with an infinite lexicon could be *learnt*.

Even if the rules *are* limited, the homonym theorist requires a competent speaker to have a knowledge of each homonym-appropriate word and its unique homonym-derivation rule (e.g. $\mathbf{and}_t$ and **(Conj)**). The type-shifting rules may be totally general, and so plausibly innate, but this cannot be said of the homonym-generation rules[120]. They only apply to some *sublexicon*, which cannot be innate. These rules must be *learnt*. However, it unclear if we *do* learn these rules when acquiring the language.

[117] Chomsky 1986 chapter 3

[118] Chomsky 1956

[119] "[T]he infinity of language is not a phenomenon in any sense whatsoever." (Collins, 2011): a language has a finite base. Only finitely many strings of the recursive model are ever realised.

[120] This also distinguished rule-generated homonym theory from type-shifting.

D. A minor worry is specifying *which* homonym occurs in an expression when calculating the meaning of a composite expression. This is a problem of *procedural effectiveness*: we want some method of determining which homonym features.

One solution is a generous interpretation. If there *is* a homonym which could generate a meaningful composite expression, we assume this homonym features. There may be doubts about the computational effectiveness of this kind of approach, see § 4.4.1.

3.2.2 Homonym Theory as Semantically Connected Families

The homonym explanation of the intuitive connection between homonyms is less satisfying than that of accounts with unified definitions of ordinary words[121]. They try to improve matters by arguing that the homonyms form 'families' which are semantically similar.

However, they may struggle to express this semantic similarity, as doing so requires the power to internally express common features between lexical entries of different types.

Suppose the system *is* strong enough to express this similarity *as a meaning*. A system which defines the term using this 'similarity' meaning would be theoretically adequate, *without using additional homonyms*. But homonym theory can *interpret* such an account, so homonym theory has a proper sub-theory which is theoretically adequate. The extra homonyms are thus unnecessary theoretical baggage.

For example, in § 2.1.1, the different **not**-homonyms are putatively connected by some *abstraction notion* of negation. If this can be properly expressed, the abstract notion has a better claim to being the 'genuine' negation than any particular homonym[122].

3.2.3 Type-Shifting and Parsimony

The type-shifter immediately avoids some of the commitments of the homonym theorist, principally the inflated lexicon, and its consequences.

Firstly, they can give a more plausible account of language acquisition than the homonym theorist. A learner does not need to encounter each homonym, nor understand the many localised homonym-generation rules. Instead, their rules are general, so speakers need only learn the small number of type-shifting rules[123]. The type-shifter could even say that the type-shifting rules are *innate*, and don't need to be learnt at all[124]. Such a position is not open to the homonym theorist: we do not have innate lexicons.

[121] Champollion 2016

[122] This 'abstract notion' seems untyped, which justifies the inclusion of semantically untyped terms (§ 4).

[123] Mix (§ 4) has no type-shifting rules for speakers to learn.

[124] Chomsky 1965 chapter 1 §8

There is also a serious reduction of lexicon size: the type-shifter is free to posit one (or two, e.g. $\mathbf{not}_e$ and $\mathbf{not}_t$ in § 2.1.1) *genuine* lexeme for each pretheoretic word in the lexicon. So there is only one (sic. two) 'basic' lexical entry and type for **and**, which is more in line with the folk lexicon.

Moreover, *even if* the type-shifter commits to a fully general derivation rule (i.e. infinitely many types), there can still be a finite lexicon.

A. A central issue for the type-shifter is the questionable axiomatic strength of their rules, which appear to make the system *too strong* for a semantic theory of natural language. Specifically, the type-shifting rules allow *implausibly many* semantic types[125] for natural lexemes. We can unpack in two ways:

1. It is implausible for natural systems, like natural languages, to have *infinitely* many types, either *for a particular lexeme* or *in total.*

2. Even a bounded hierarchy allows for type-shifts which are not witnessed in natural language[126].

Concentrating on 1., the rules described above (e.g. **(M)**, **(G)**) are *totally general.* Each time a type-shifting rule is applied, the type complexity increases, just as with the derivation rule.

Recall from § 1.6.2 that a fully general *derivation* rule is unconvincing, as it generates an *infinite* hierarchy of types, which cannot be necessary for an account of a *natural* system. The same can be said about the type-shifting rules: they generate an infinite hierarchy of possible types for each lexeme to which they apply. For example, a type (a, b) lexeme also has a meaning of type $((c, a), (c, b))$ for all of the infinitely many types c. Not only does this mean the *system* must contain infinitely many types (as they are witnessed by particular shifted lexemes), but also that *particular words* have infinitely many meanings.

The type-shifter can simply reply that, exactly similarly to the derivation rule, we can legitimately restrict the number of uses of type-shifting rules.

More troublingly is a version of issue 2. Even if the rules are limited in *number of uses*, there is no limit to *which* words we can apply the rules: they are general rules for *all* types. So, for example, **(G2)** can raise an intransitive verb from type (e, t) to $((e, e), (e, t))$. This is the type for lexemes which, intuitively, take adjectives[127] to intransitive verbs. However, this description seems totally disconnected from the meaning of an intransitive verb, for instance the following non-exotic expression yields a truth-value:

(36) *John big walks.

[125] In a (flexibly typed) Lambek calculus, the number of logically non-equivalent meanings for an expression is finite (GAMUT 1991 §7.3.2). However, this still allows for unnatural readings, and for lexemes to have infinitely many different meanings.

[126] Lewis 1972

[127] See § 2.5.

Not only is *this particular* type-shift semantically illegitimate, it seems *all cases* of **(G2)** applied to transitive verbs are illegitimate in English, there are *no natural language examples.*

Of course, the defender might plausibly argue that the failure of (36) is *syntactic.* We work in an ambient syntactic theory. Presumably, some rule of English syntax forces adjectives to precede the NP they modify, invalidating (36). The fine detail depends on our theory, but, generally, there is no valid syntax tree for (36).

One class of solution is to weaken the shifting rules, explored in § 3.2.4. We note only that restricting the rules *by type* may be unsatisfactory, as their generality was in key the solution to the type ambiguity of connectives (particularly **(G)** in the case of **not**).

An alternative, more general, response to issues of formal strength is the introduction of a *meta-rule* which restricts *the use* of the type-shifting rules:

(Nec) Only apply type-shifting rules when necessary.

We investigate this carefully in § 4.4.1.

The general suggestion here is that the rules governing type-shifting proposed by the type-shifter are simply *too strong and general* to be rules governing the natural system of natural language.

B. Type-shifting also has questionable computational effectiveness. Firstly, type-shifting allows for non-terminating patterns of shifts in a calculation, i.e. calculation loops and unbounded sequences.

Unbounded sequences might occur when the 'wrong' expression shifts. For example, expression a might need to be type-shifted before it composes with b. As there is no optimal way to apply the type-shifting rules, these shifts could become unsynchronised, where b shifts up first, which would then not compose, followed by shifting a, etc.

Secondly, type-shifting has some *calculation ambiguity*, as there is no clear general algorithm to determine which type-shifting rule to use at a given stage (see § 3.2.4).

Again, **(Nec)** may provide a solution. For example, loops are unnecessary for a calculation, so are delegitimised (see § 4.4.1).

C. Perhaps most importantly, type-shifting is total, so must assign *some* type to the problematic lexemes (or one per homonym). However, any such choice is entirely arbitrary. The data suggest that these lexemes do not comfortably fit into the hierarchy of types. Worse, the assignment is apparently based on reducing type complexity rather than on *the meaning of the term itself.*

For example, **and** is assigned type $(e,(e,e))$ or $(t,(t,t))$, as these are the least complex types of Basic which characterise *some* uses of **and**. But this is an affectation. **and** is not 'at heart' entity-conjunction any more than any other kind of conjunction. The type-shifting rules supposedly "map arbitrary denotations of a basic type to secondary denotations of the appropriate non-basic type"[128] but there is *nothing secondary nor non-basic* about the other uses of **and**.

Such arbitrary type assignments are unnecessary, see § 4 and § 4.5.5.

3.2.4 Restrictions: More Parsimonious Type-Shifting?

One possible response from the type-shifter is to *restrict* the rules from their *total generality*. We examine some candidate restrictions.

The first candidate restricts to a particular sublexicon:

(M↾S) $a \mapsto \frac{c}{c/a}$, but only applied to sublexicon S

If S is finite, this restriction adds only finitely many new lexical entries and types.

However, it seems to defy the whole project of compositional semantics *without* semantic filtering (§ 1.4). Rules restricted by lexicon are non-general, requiring prior lexical knowledge to generate the semantics of composite expressions, rather than concerning the types of subexpressions alone.

Perhaps worse, it is unclear how such a restriction could be a *natural rule* of a *natural system*: lexicons are not innate, so **(M↾S)** must be language specific and seriously ad hoc.

This also seems virtually identical to rule-generated homonym theory.

Rules are more plausibly limited by *type-form*:

(M↾F) $a \mapsto \frac{c}{c/a}$, but only for types of forms from F

These resemble restrictions based on *number of uses* of each rule (see § 1.6.2). In fact, type-form limitations subsume numerical limits, as particular rule application iterations correspond to particular type-forms.

However, type-form limitations do not have the simple form of a theorem. Compare:

(M) $a \mapsto \frac{c}{c/a}$ for all c.

(M') $a \mapsto \frac{c}{c/a}$ for any c except for in cases of the following forms ...

[128] H & K 1998 §4.2.2

(M') is clearly more complex than **(M)**. Even worse, there is no prima facie reason why the set of forms need be finitely describable. This suggests that this restriction is ad hoc.

(M') is *overly* specific. It appears to be a contrived *guideline*, which is specifically designed for a handful of problematic phenomena, rather than a natural *rule* governing a natural system. It lacks the requisite generality we expect from such a rule. It is ad hoc and syntactically complex to the point of inexpressibility.

Further, these restricted rules *do not* have the same mathematical motivation. We motivated (some of) the rules by treating the type system as a genuine algebra, where these rules are *theorems*. However, these theorems were characteristically *un*limited. **(M')**, **(M|F)**, etc. *do not* represent theorems, and hence do not have good mathematical motivation.

Here is the dilemma: either the rules are general, so they have too much formal strength (meaning they generate unsuitable semantics), or they are overly specific, so lose their (conceptual and mathematical) naturalness and are ad hoc instead.

It seems that the type-shifter must resort to a restrictive *meta-rule* like **(Nec)**, allowing the type-shifting rules themselves to be general and well motivated, but protecting against the problem cases associated with great formal strength.

3.3 Controversial: Competent Speaker Intuitions, or Informal Semantics

We must have *some* data which we can use in our semantic theorising. This presumably includes truth value judgement tests, where competent speakers are asked to classify a *sentence* as true or false in a particular scenario.

Possibly these truth value judgement tests are the *only* semantic evidence we have. This might be justified by the apparent opacity of our (possible) Chomsky-Fodor language modules[129]. We cannot easily introspect to gather semantic data, the module is somehow obscured.

Even if there *are* intuitions about the semantic structure of non-sentence expressions, it is not at all clear that these are more reliable than they would be in other empirical enquiries. We do not give evidential credit to the 'hunches' of suitable speakers in the case of e.g. *chemical composition* of some substance.

Despite this, I claim we *do* have some kind of non-sentential semantic evidence with which we can tentatively theorise. We seemingly have intuitions about the *similarities* in meaning between different expressions. This seems sufficient to lend a little naturalistic support to the inclusion of untyped expressions in our formal semantics.

[129] Fodor 1983

3.3.1 Intuitions about Types

Here is a controversial possibility: competent speakers *do* have (implicit) intuitive or everyday data about the semantic *similarity* of words. It seems as though competent speakers, and historical theorists of various kinds, *categorise expressions by kinds of meaning*.

Some supporting evidence comes from the fact that differing traditions of historical and naïve studies of language and grammar *are* categorical[130] (e.g. the Western European[131] and Islamic[132] traditions).

Moreover, the *particular* categories are often conceptually similar: there are verb-like categories and noun-like categories. Words seem to be classified by *both* where they fit in expressions (i.e. syntactically) *and* what they *mean*. We think that **bake** is more like **prepare** than it is like **lake**, despite their similar phonology and orthography. Not only do the two verbs fit 'in the same places' in expressions, they also have a common being-about-objects.

Even when staying in the realm of 'concrete' language, any putative data becomes much more opaque when moving beyond nouns and verbs. Semantic intuitions or folk definitions are foggy in relatively simple cases of functional-type expressions, like adverbs and adjectives.

Even so, we appear to categorise syntactically *and* semantically, to some extent.

3.3.2 Intuitions about Untypedness

Here is another controversial possibility: the same ordinary definitions and intuitive data suggest that there is something fundamentally *dissimilar* about connectives compared to other parts of language.

Dictionary definitions and (putative) intuitive meanings for nouns are *things in the world*. Similarly for verbs, the data suggest a meaning like 'something that is *done*', or a process. Even if fine-grained intuitions about *particular* meanings are dubious, we can intuitively *classify* the expressions together by meaning. However, the data suggests the connectives do not fall into these groups[133].

Instead, any (uncertain) intuition about connectives seems more *abstract*, and less about particular parts of reality. For example, **or** describes alternatives, options. But there is no intuition about the *kinds* of the options in question. **and** is exactly similar. Indeed, this seems to rationalise the join-like definitions in some formal semantics[134], suggesting the overlapping of two previous notions, without committing to particular *kinds* being joined (see § 4.2.2). We can

[130] This might be platitudinous: almost every analysis can be interpreted as categorical.
[131] von Humboldt 1836, §21
[132] Al-Mansour 2001
[133] The principal definition of **and** in the OED is conjunction *undistinguished* by such categories (Oxford Dictionaries 2018).
[134] Hendriks 1993

characterise this abstractness by saying that these intuitions never makes use of types!

Hence, there seems to be some *natural* justification for the thesis that certain parts of natural language are untyped, beyond the formal niceties explored in § 4.

3.3.3 Criticisms

A. The critics might deny that this is evidence of genuinely *semantic* intuition, instead rationalising the observation as intuitions about *syntactic categories*. Syntactic data collection for non-sentence expressions is much more standard. Linguists typically ask about the grammaticality of expressions of various kinds, so "the big and round ball" might be judged grammatical. However, collecting *semantic* data from ordinary competent speakers is treated with much greater suspicion.

B. A further worry here is that any putative intuition about 'abstractness' ostensibly extends beyond the connectives. For example **that**, **the** and **is** also appear suitably abstract. Each is also (purportedly) type ambiguous. **the** has similar issues to the quantifiers[135], and the **is** of identity is considered 'syncategoremic'[136], like the connectives (§ 2). Should we accept this as evidence that these words are untyped?

There are alternative explanations. **that** seems to be *lexically* ambiguous, not type ambiguous (§ 2.6). Indeed, both the copula use of **is** and the relative-clause use of **that** may be semantically vacuous[137]. Revising the claim, we might say that intuitive or ordinary data about an abstract meaning does not *necessitate* untyping, but is correlated with it.

[135] See footnote 53. Referentialists about definite descriptions might dispute this.
[136] GAMUT 1991
[137] H & K 1998.

4 Proposal: Mixed System, Mix

Here we introduce a possible alternative account, Mix. Mix is mixed in the sense that the language allows for both typed and untyped terms[138]. So a lexicon for a particular language might still have no untyped terms. Ideally, this would only occur if the language had no such type-ambiguity cases.

4.1 Structure of Mix

Structurally, Mix is similar to Basic. The typed part has exactly the same syntax and semantics. We have a type system, M:

Mix 1 (Ontology)**.** *The foundational types of Mix are* $e \in M$ *and* $t \in M$.

Mix 2 (Restricted Derivation)**.** *If* $c, d \in M$ *have total complexity*[139] *lower than* α*, then* $(c, d) \in M$.

Mix 3. *If* $(c, d) \in M$ *then* c, $d \in M$.

Mix 4 (Bi-Application)**.** *An expression of type* $c \in M$ *applied to a type* $(c, d) \in M$ *yields type* $d \in M$ *in either of the following ways:* $c{\cdot}(c, d) \mapsto d$ *or* $(c, d){\cdot}c \mapsto d$.

Mix 5 (Semantic Denotation Domain)**.**
$\boldsymbol{D}_{e,D} = D$
$\boldsymbol{D}_{t,D} = \{0, 1\}$
If $c, d \in M$*, then* $\boldsymbol{D}_{(c,d),D} = \boldsymbol{D}_{d,D}^{\boldsymbol{D}_{c,D}}$

For *typed* terms, Mix can be semantically parasitic on Basic, with the meanings of the typed lexemes being exactly their meanings in the totally typed system, i.e. suitable elements from the appropriately typed domain $\mathbf{D}_{c,D}$. When the lexemes have the same type in the two systems, they have the same lexical entries. Presumably, this holds for **ball**, **Mary**, **aggressively** etc.

Mix 6 (Untyped Elements)**.** *Some elements are untyped, denoted* u.

One cannot derive further types from untyped elements. For example, there is no such 'type' as (u, t).

Mix is immediately more parsimonious than the flexible typer: it stipulates no additional homonyms, nor any type-shifting rules.

The compound expressions containing untyped elements are determinately typed, so the untyped elements are somehow 'sensitive' to the types of elements around them. For example, [**John and Mary**] features the untyped **and**, but is type e.

However, these rules together are not 'complete': we do not have an application rule for untyped elements, to explain this sensitivity to surrounding

[138] Some accounts are *entirely* untyped, e.g. untyped lambda calculus, Barendregt 1984.
[139] See footnote 108.

expressions. A calculation initially featuring an untyped term, features it in all further steps. This cannot be deemed theoretically adequate.

To form a theoretically adequate account, we explain how the untyped terms are used in calculations.

Before this, we describe the *semantics* of the untyped terms.

4.2 Semantics of Untyped Expressions

What does it mean to be untyped? u doesn't obviously designate a natural ontic category, unlike the other basic types (entities, truth-values, events, etc.). This is unsurprising: being untyped is the *absence* of a relevant type. However, this 'absence' description is unsatisfactory, we owe an account of the *meanings* of the untyped terms.

4.2.1 Intersectional Theory of 'and'

The connectives are the main class of possibly untyped terms. We focus on explaining **and**, as a typical example thereof. In our diversion into event semantics, we gave two possible definitions for **and**, which were based on the notion of intersecting various functions (with ancestor t). These were:

1. $\lambda f_{(e,t)}\lambda g_{(e,t)}\lambda x.x \in \{x' \mid g(x') = 1 \wedge f(x') = 1\}$
2. $\lambda f_{(s,t)}\lambda g_{(s,t)}\lambda e.e \in \{e' \mid g(e') = 1 \wedge f(e') = 1\}$

These meanings are *dissimilar*, as the initial variables are differently typed. However, there is a clear pattern here. An obvious candidate *untyped* meaning is available:

$$[\![and]\!] = \lambda u\lambda v\lambda o.o \in \{o' \mid v(o') = 1 \wedge u(o') = 1\}$$

Where u,v, and o are untyped variables. We do not state that o is the argument of u and v, as this is forced by the definition (e.g. $v(o') = 1$).

The claim is that the parallel definitions are adequate for the other connectives. The parallel definition for **not** is:

$$[\![not]\!] = \lambda u\lambda o.o \in \{o' \mid u(o') = 0\}$$

This clearly works in most cases. The story for predicates may be more complicated. One generated meaning of "John does not walk." is 'John is the agent of a not-walking event', rather than 'John is not the *agent* of a walking event'. However, proper attention to the syntactic structure generates the required meaning[140].

[140] Champollion 2014(b)

In general, it is seemingly difficult to avoid at least *some* ambiguity for **and**, e.g. the 'summative' use ("two *and* two is four"), the ordered n-tuple use[141], genuinely elliptical uses[142]. This is unlikely to be comprehensive. Plausibly, there is a pragmatic explanation for some of these (the maxim of manner might explain ordering and ellipsis[143]). If not, we might accept that *not all* uses of **and** have the same meaning.

However, there can still be one *principal* meaning of **and**. If these aforementioned uses of **and** are genuinely different meanings, then they seem non-standard, and less prevalent than the common or garden usage of **and**. Moreover, there are no obviously different, non-standard uses of the *other* connectives, e.g. **but**. So, the untyped, intersective definition of **and** above is a substantive improvement, being both elegant and highly general, as are the equivalent definitions for other natural connectives.

Further, given that an adequate account of semantics must explain these non-standard uses of **and** *anyway*, the untyped definition is more unified than the alternative accounts. Mix does not carve up different meaning of ordinary **and** like the type-shifter, nor does it stipulate additional homonyms.

We now focus on this principal, ordinary use of **and**. The candidate intersective definition of **and** is highly effective when used in the great majority of *functional* cases (with ancestor t). However, it is prima facie less satisfactory in two other ordinary cases:

1. Sentential conjunction.
2. Uses of **and** with ancestor e, most critically, *entity* conjunction itself.

The candidate definition doesn't apply in non-functional cases, so says nothing here.

Event semantics provides no help in accounting for sentential conjunction. Here, the denotation of a sentence is some collection of events, e.g. $\lambda e.(agent(e) = Mary \wedge ...)$. Seemingly, the meaning of "ϕ and ψ" is the collection of all ϕ-events *adjoined to* the collection of all ψ-events, *not* the events which witness each of ϕ and ψ independently. This suggests a non-intersective, *collective* meaning of **and**.

However, conjunction manifested as intersection is the standard conception of conjunction in possible world semantics[144], which means the unified untyped definition of **and** again seems sufficient[145]. In possible world semantics, the

[141] Link 1998
[142] See footnote 72
[143] Grice 1991 §2.
[144] Kripke 1963
[145] This requires an extra (wide-scope) operator in the structure of a sentence which converts an event semantic value to a possible world semantic value. But it seems as though some suitable conversion operator could be defined.

denotation of a sentence is a collection of possible worlds, specifically, the ones in which that sentence holds. Here sentential conjunction naturally corresponds to the *intersection* of the collections of possible worlds, i.e. those worlds which are both ϕ-worlds and ψ-worlds. In which case, the *untyped, intersectional* definition of **and** remains totally adequate!

4.2.2 NP Conjunction and the Collective Theory of 'and'

Ancestrally-e conjunction is more troubling. The candidate intersectional definition *could* be applied in the entity conjunction case, given a Montagovian account of proper names (as type $((e,t),t)$ § 2.5.2).

However, this approach is unpalatable. Not only are there independent worries about this type assignment (§ 3.3), the generated meaning seems unsuitable.

(11) John and Mary walk.

If the meaning of **John** uniquely picks out John (even if it is type $((e,t),t)$), and the same for **Mary**, their *intersection* seems unable to generate the *collective* meaning we expect from (11). Indeed, the intersection is either *empty* (nothing is both exactly John and exactly Mary), or is something like 'the set of common properties of John and Mary', which doesn't obviously constitute a *conjunction* of Mary and John. Assuming they are distinct, this common set does not 'add up' to even one of them, as it were.

Further, the candidate intersective definition of **and** seems unable to cover some instances of common noun conjunction[146].

(37) Every doctor and lawyer has a degree.

(38) Ten left shoes and right shoes were put in pairs.

As with proper names, the *intersection* is not the required meaning here. Instead, **and** means some kind of *fusion* or even a *union*. So, **and** would have the following meaning:

$$[\![and_{(e,t)}]\!] = \lambda f_{(e,t)} \lambda g_{(e,t)} \lambda x.x\{x' \mid g(x') = 1 \vee f(x') = 1\}$$

Worse still, this appears to be the *standard* conjunction for common nouns[147].

Can we unify these *collective* meanings with our elegant *intersective* candidate into a single (principal) definition of conjunction? Or, must we concede that **and** is semantically ambiguous?

[146] Champollion 2016

[147] Champollion (2016) discusses possible intersective examples. "Every person who is (both) a doctor and a lawyer has a degree" cases are non-problematic: conjunctions of *defined* NPs (type $((e,t),t)$ are expected to be intersectional.

One problem with unification via sentential conjunction is that (37) *might* have a meaning-preserving elliptical paraphrase[148], but (38) does not.

(37') Every doctor has a degree and every lawyer has a degree.

This adequately paraphrases (37), given a suitable account of sentential connectives. However, this elliptical strategy is not powerful enough to explain (38). The equivalent paraphrases is:

(38') Ten left shoes were put in pairs and ten right shoes were put in pairs.

Champollion claims that the intersective definition can account for collective examples like (38), and so is unifying[149]. He suggests that collective sentences rely on three silent operators: Raising, Intersection, and Minimization. The strength of this approach is an independent question. I mention two problems he faces. Firstly, the evidence for the silent operators is questionable (see his §3). Secondly, Raising seemingly requires a universal set in the background model theory, which is dubious.

Instead, to explain these collective uses of **and**, we consider a radically different approach: *fusion*. [**Mary and John**] is not the *intersection* of Mary and John. Rather, it is a *mereological fusion* of the individuals, something like 'Mary and John, considered together'. (37) and (38) seem exactly similar. This is *fusion manifested as disjunction*, as in the definition $[\![and_{(e,t)}]\!]$, above.

Link formalises this fusional meaning of **and** with his algebraic semantics, developing into a lattice-theoretic analysis of plurality[150]. He defines two kinds of sums of a pair of objects, a and b:

> "$a + b$ is ... the material fusion of a and b; $a \oplus b$ is the *individual sum* or *plural object* of a and b."
>
> *Link 1998*

He argues that these are *not* identical. Though $a + b$ *constitutes* $a \oplus b$, $a \oplus b$ is a metaphysically more structured object[151].

[148] This elliptical strategy is itself dubious, as it requires *determiner* ellipsis (not just VP-ellipsis), to elide **every**. This seems to overgenerate, cf. "Every woman is happy and [every] man is happy too.".

[149] Champollion 2016, see also Winter 2001 §2.3.3.

[150] Link 1998

[151] $a \oplus b$ maintains some part-structure, with each component retaining individual existence and distinguishability. This is not true of $a + b$.

We are concerned with the basic entailment relationship between collections and their members[152], and hope to explain several kinds of (non-)implications:

(39) John and Mary walk. ⇒ John walks and Mary walks.

(40) John and Mary walked together. ⇏ John walked together and Mary walked together.

(41) Every doctor and lawyer has a degree. ⇒ Every doctor has a degree and every lawyer has a degree.

(42) The water in my cup evaporated. ⇒ The water at the bottom of my cup evaporated.

In general[153], **together** enacts a difference in a predicate, p. Whilst p may be suitable for unstructured arguments ('cumulative'), like $john + mary$, [p **together**] holds only of the more structured collective entity, like $john \oplus mary$.

So Link's difference between (39) and (40) is that **walk** is cumulative, it can take either $john + mary$ or $john \oplus mary$ as subjects[154]. However, [**walked together**] requires the plural subject, so *only* takes the subject $john \oplus mary$.

Exactly similarly to this entity fusion, in (41), the meaning of [**doctor and lawyer**] is, roughly, the fusion of the relevant denotations. Link gives distinctively collective definition of common noun (e,t)-conjunction, $P \oplus Q = \lambda x(Px \vee Qx)$, in agreement with $[\![and_{(e,t)}]\!]$ above[155].

Whilst examples like (42) are not our principal concern, Link's account can uniformly explain the 'explicit' plurals of (39)-(41) *and* the mass terms in (42). He argues that water is the material fusion of some parts, and **evaporated** takes material fusions as arguments. As all the water in the cup evaporated, the water at the bottom (a part of the material fusion) also evaporated. This gives reasonable semi-independent motivation for his account.

So, Link's analysis is that, in (11), **and** has the meaning of *forming a plural subject*[156] from the two constituents, at least in the case of NPs. The plural subject $john \oplus mary$ walked together, or met, but the *material sum*, $a + b$, does not[157].

Given this analysis, one possible proposal is that *the* only genuine NP conjunction is the plural sum. This would permit an unambiguous definition for **and**, where $\lfloor c \rfloor$ is the type of c and u,v, and o are untyped:

[152] Champollion 2014(b)
[153] Cf. **together**, [**danced together**]
[154] Link 1998 §2.4.6.
[155] Link 1998 §1.2
[156] "Collective entity", Champollion 2014(a)
[157] Verbs need not be subcategorised into those that take plural sums as arguments, and those that take material fusions. If $a + b \neq a \oplus b$ then, possibly $p(a + b) \neq p(a \oplus b)$. $mary + john$ and $mary \oplus john$ are two different objects to which [**walked together**] might apply.

$$\begin{aligned}[[and]] = \lambda u\lambda v\lambda o.(\ &(\lfloor u\rfloor = \lfloor v\rfloor = e \rightarrow && u \oplus v) && \wedge\\ &(\lfloor u\rfloor = \lfloor v\rfloor = (e,t) \rightarrow && o \in \{o' \mid v(o') = 1 \vee u(o') = 1\}) && \wedge\\ &(\lfloor u\rfloor = \lfloor v\rfloor \notin \{e,(e,t)\} \rightarrow && o \in \{o' \mid v(o') = 1 \wedge u(o') = 1\}))\end{aligned}$$

However, there is a problem with the (e,t) part of the definition here, we want some *intersective* cases of (e,t)-conjunction, specifically *predicate*-conjunction (e.g. (12)). This suggests a final revised definition:

$$\begin{aligned}[[and]] = \lambda u\lambda v\lambda o.((\lfloor u\rfloor = \lfloor v\rfloor = e \rightarrow && u \oplus v) && \wedge\\ (\lfloor u\rfloor = \lfloor v\rfloor = (e,t) \rightarrow && (o \in \{o' \mid v(o') = 1 \vee u(o') = 1\} && \vee\\ && o \in \{o' \mid v(o') = 1 \wedge u(o') = 1\})) && \wedge\\ (\lfloor u\rfloor = \lfloor v\rfloor \notin \{e,(e,t)\} \rightarrow && o \in \{o' \mid v(o') = 1 \wedge u(o') = 1\}) &&)\end{aligned}$$

This supplementary definition 'mops up' the issues surrounding ordinary NP-conjunction. From one perspective, it is unambiguous and determinate, as the type of the conjuncts fixes which part of the definition is used, though there are 'options' in (e,t)-conjunction. It is also *general*, as it covers all the cases we expect it to, including in event and possible world semantics[158].

This definition is intended to be *supplementary*, extending the candidate elegant definition above. Even so, one might criticise it by saying that what it gains in generality and adequacy it loses in elegance. It is seemingly less unified than the original candidate, as it has a disjunctive, case-by-case form. The critic might insist that the three or four 'parts' of the definition represent separate meanings.

Even so, this definition is still a major improvement on homonym theory and type-shifting, which stipulate numerous meanings, and must distinguish the collective and intersective meanings *anyway*[159].

Link's alternative is non-unification[160]. He carves up entity conjunction into three or four homonyms, principally $\mathbf{and}_1$ for (roughly) expressions with a sentential paraphrase, e.g. (38), and $\mathbf{and}_2$ for plural sums[161], along with a uniform treatment for other kinds of conjunction.

However, this approach is highly reminiscent of homonym theory, and is similarly less elegant and general[162]. Further, it suffers from a major issues of semantic similarity:

[158] If we do not accept type-shifting, we may need *further* components to explain asymmetric conjunctions (e.g. e and $((e,t),t)$, as in § 2.5.2). However, asymmetric cases are still *collective*, so this is not a radical change from the suggested definition. A similar revision is possible in definition of **or** in § 4.2.3.

[159] Collective uses may be more wide-spread, e.g. "Pick up every blue and green ball." is apparently disjunctive, referring to balls which are blue *or* green.

[160] Link 1998 §3.1

[161] There is also $\mathbf{and}_3$ for ordered pairs.

[162] Winter 2001 §1.4

> "[Multiple lexical entry theory] has nothing to say about the intuitive connections between the two uses [of **and**]... it does not capture the way these uses are tied together across languages. English uses the same word for intersective and collective conjunction... this situation appears to be typical"
>
> *Champollion 2016*

Either way, these different uses of **and** are *independently* occurring linguistic phenomena, and must be explained by *any* alternative semantic theories.

Homonym explanations suffer from Champollion's criticism, but this is subsumed into the *general* semantic subcategorisation for which the homonym theorist is criticised.

The type-shifter is free to give a fairly similar definition of **and**. However, if Mix and type-shifting *do* give the same definition, the type-shifter must still arbitrarily assign a base type to **and** (e.g. $(e,(e,e))$, whilst the meaning of the word does not suggest any such type is privileged.

Most significantly, we have given a *main* definition for **and** in Mix, the intersective candidate $[\![and]\!] = \lambda u \lambda v \lambda o.o \in \{o' \mid v(o') = 1 \wedge u(o') = 1\}$. This covers a large majority of cases (it is general), and it is clearly elegant and unified. The equivalents in other accounts are apparently less general.

4.2.3 Other Connectives

A strength of Mix is that it allows for a fairly uniform account of the connectives, *including* the distinctively natural connectives, like **but**, **as well as**, etc.

Whilst some exclude the 'logical' connectives from the ordinary lexicon, giving each a *particular rule* governing its usage[163], Mix unites the natural and logical connectives. For example, **not** is considered difficult to analyse, but our earlier definition[164] does much of the requisite work, e.g. in verbal and sentential negation:

$$[\![not]\!] = \lambda u \lambda o.o \in \{o' \mid u(o') = 0\}$$

This does not obviously explain the controversial[165] case of entity negation. If necessary, **not** can be defined disjointly, as with **and**.

or is easier to tackle, as it is less semantically ambiguous than **and**[166]. $[\![or]\!]$ *is not* the dual of $[\![and]\!]$, i.e. a 'split' definition as the union of functions and

[163] Partee & Rooth 1983
[164] Similar to Champollion 2014(b).
[165] See footnote 68.
[166] There is still exclusive-inclusive ambiguity.

intersection of entities. Instead it seems more uniform[167], with $\mathbf{or}_e$ acting like a disjunctive union, perhaps defined like so[168]:

$$[\![or]\!] = \lambda u \lambda v \lambda o.(\ (\lfloor u \rfloor = \lfloor v \rfloor = e \rightarrow u \vee v) \qquad \wedge$$
$$(\lfloor u \rfloor = \lfloor v \rfloor \neq e \rightarrow o \in \{o' \mid v(o') = 1 \vee u(o') = 1\})$$

Parallel definitions for (other) natural connectives are plausible. Except for **and**, the connectives are not so multiply semantically ambiguous. So the elegant definitions in Mix are unified and general, unlike the definitions in totally typed systems.

4.2.4 Untyping Beyond the Connectives

Type ambiguity motivated the introduction of untyped elements into the formal system. Modifiers may also suffer from type ambiguity. So one might suggest these as candidates for an untyped definition.

We sketch a non-Montagovian[169] untyped explanation of adjectives. I do not defend this position, it is simply *within the formal power* of Mix, and displays the kind of moves available in explaining other putatively untyped phenomena. It is a style of explanation, which is implausible here, but may be plausible for *other* classes of lexemes.

They can assert that adjectives are untyped, which explains the requisite calculations, without stipulating type-shifting or homonyms. For example:

(31) big John

(32) big dog

The plausible types for **big** were:

(31) (e, e)

(32) $((e, t), (e, t))$

Given some untyped definition of **big**, we can calculate like so:

(31) $u \cdot e \mapsto e$

(32) $u \cdot (e, t) \mapsto (e, t)$

This 'solution' is a double-edged sword, with two major problems.

[167] Champollion (2016) claims the *only* definition is uniform and functional: $[\![or]\!] = \lambda u_{(c,t)} \lambda v_{(c,t)}.u \cup v$, exactly dual to his $[\![and]\!]$.

[168] Roelofsen (2016) discusses this further.

[169] See § 2.5.2.

Firstly, it is implausible that adjectives are untyped. One argument for having *any* untyped terms was that the connectives seem different from other parts of language (§ 3.3). They are somehow abstract, and 'devoid' of metaphysical, non-logical meaning. However, this *cannot* be said of adjectives. Indeed, they are typical of ordinary language, simply modifying NPs. It is prima facie odd for adjectives to be more like *connectives* than like other modifiers, or the (typed) nouns themselves. It is also implausible that an ad*verb* is typed whilst its accompanying adjective is untyped[170].

Secondly, if **big** *really is* untyped, we can use it in other calculations which *are not* permitted. For example, [**big runs**] would be a legitimate, well-formed expression of type (e, t)[171]. Untyping adjectives is simply too powerful.

Mix is unapologetically tailored for explaining connectives, so it may not be flexible enough to explain modifier type ambiguity in English (though a Montagovian approach would suffice). However, it is still possible that other type ambiguity *could* be suitably explained by untyping, perhaps outside of English. The explanatory power of Mix can extend beyond explanations of natural language connectives.

4.3 Is u a Disguised Type?

Being untyped does not seem to carry any *strong* semantic content, as e and t did. Types give more information about the term (principally, the *kind* of meaning it has), which being untyped does not.

Nevertheless, a critic might claim that u is a 'disguised' type. The original presentation of the ontology could be misleading, they might say. Instead, the 'real' underlying system is, for example, a variant of Basic with an ontology enlarged by u.

One response reiterates that being untyped is a *purely formal* property. In Basic, 'entity' and 'truth-value' are assumed to correspond to their ordinary meanings, so the particular type of an expression is semantically informative. But 'untyped' does not have an ordinary (semantic) meaning in this way, so there is no possible analogue for u!

Moreover, u functions too different to be considered part of the ontology. For example, the action of derivation rules on types is a major feature: from a pair of types, one can derive a new type[172]. Derivation from c to (c, t) is particularly central for a semantic category c. However, derivations involving u are semantically unintelligible, instead u sits 'outside' the derived type hierarchy.

[170] Similarly for the linked noun: **quiet** being untyped whilst **quietness** being type e is implausible.

[171] A syntactic restriction might exclude these cases, see § 3.2.3

[172] This is reminiscent of primary school grammar, where a predicate is 'a sentence missing a noun(s)'.

Further, the genuine types have a unified application rule, **(A)**, which does not apply so simply to untyped terms. Instead, calculations involving untyped terms require us to look at their lexical entries (and perhaps use **(Untyped)**).

4.4 Calculating Using Untyped Terms

For the descriptive adequacy of Mix, calculations featuring untyped terms must be explained. The calculations featuring connectives were simple. There was an 'obvious' way to compose the untyped meaning with the conjuncts, especially so in the supplementary, disjoint definition of **and** (§ 4.2.2), where the type of the conjuncts explicitly determined which part of the definition occurs in that composite meaning.

Recall the example:

(7) Most babies don't cry.

To generate the composite meaning, we need a way to calculate like so:

$$(e,(e,t))\cdot u\cdot(e,t)\overset{*}{\mapsto}(e,(e,t))\cdot(e,t)\mapsto t$$

Much of the project of flexible typing boils down to legitimating steps like $*$ (they do not talk of untyped elements u, but the natural translation is clear).

In (7), the syntactic structure dictates that the untyped **not** composes with the verb. So we can see the obvious (type-compatible) composition in this case:

$$\lambda u\lambda o.o\in\{o'\mid u(o')=0\}\cdot\lambda x.\textsc{cries}(x)\longmapsto\lambda x.x\in\{x'\mid\textsc{cries}(x')=0\}$$

The calculation then continues exactly as in total typing, and generates the expected meaning.

In general, the process by which untyped terms are used in calculations is just the same. Suppose we have a well-formed expression, which has a binary branching structure, featuring an untyped term. If the untyped term is the daughter of a node, then there is some type which the untyped term can 'act as'[173] in order to compose, i.e. there is some *type-compatibility* between the daughter node terms. So, we can generate the intuitive compositional semantics of these expressions[174]. Hence Mix is descriptively adequate.

This is precisely how the total typer calculates[175]. We look at the nodes in the syntactic tree, and then look up the lexical entries in the lexicon for each daughter-term. If, for each term, there is a (suitable) lexical entry which

[173]'Acting as' type c means that the calculation step is equivalent to composition by an element of type c. For example, in intransitive VP-conjunction, **and** acts as $((e,t),((e,t),(e,t)))$.

[174]Expressions featuring *multiple* untyped lexemes are uncomplicated. Consider "Mary but not John walks." (§ 2.1.1). We must legitimate a calculation: $e\cdot u\cdot u\cdot e\cdot(e,t)\mapsto\cdots\mapsto t$. Syntactic structure explains this: we analyse the *smallest* subexpression featuring an untyped lexeme first. We thus calculate: $((e\cdot u\cdot(u\cdot e))\cdot(e,t))\mapsto((e\cdot u\cdot e)\cdot(e,t))\mapsto(e\cdot(e,t))\mapsto t$.

[175]H & K 1998 §3.

composes, we compose those meanings. Expressions which feature no untyped terms are analysed *exactly* as in total typing.

Perhaps pragmatic and contextual factors contribute in determining which entry is suitable, but no more so than is otherwise necessary. There is a *solid core* which does not rely on pragmatic information, and is just as algorithmic as the totally typed calculation process. This core certainly contains the less ambiguous connectives, e.g. **but**, **not**. If the term *is* semantically ambiguous (e.g. **and**), the following rule characterises the choice of lexical entry for use in a calculation:

(Untyped) If a daughter node, u, is untyped, calculate the meaning of the composite expression using the type-compatible lexical entry for u.

4.4.1 Generous Interpretations of Mix and Flexible Typing

One might complain that **(Untyped)** is not obviously procedurally or computationally effective, i.e. we cannot write down an explicit algorithm for calculating the meanings of composite expression in genuinely novel calculations (outside of the connectives).

Put another way, calculations in Mix apparently require a privileged epistemic position to determine a type-compatible use of an untyped term (i.e. it requires *semantic filtering*).

The computational effectiveness of Mix is justified by a direct comparison between its computability and that of the flexible typers. Each account seemingly requires a generous interpretation for the selection of suitable lexical entries and the use of the system rules. However, greater generosity is needed for flexible typing than for Mix.

Both homonym theory and type-shifting required *strong background assumptions* to improve their plausibility, whilst maintaining their additional structure (longer lexicon and more rules respectively). Meanwhile Mix does not require such additional structure.

To bolster the flexibly typed accounts, advocates restricted their formal strength. The most convincing type-shifting variant used a background meta-rule[176]:

(Nec) Only apply type-shifting rules when necessary.

There is no determinate, algorithmic process characterising the use of **(Nec)**, as their proviso 'necessary' is left unpacked. Meanwhile, accusations of the procedural ineffectiveness of Mix are vague and have little bite.

[176]Similarly, homonym theorists must stipulate *exactly which* homonyms are necessary. Rule-based homonym theory requires an exact analogue of **(Nec)** to limit the number of homonyms generated. See § 3.2.1.

One might restrict 'necessity' by only allowing type-shifts when the existing types do not compose. But even this does not generate a determinate process. There is no general reason to think that there is a *unique* type-shift that yields composable subexpressions, nor to apply one rule before another. So there is no optimal algorithm for applying the rules to constituents, meaning **(Nec)** still allows for loops and unbounded sequences, e.g. the example in § 3.2.3 **B** is non-terminating under the 'existing types do not compose' algorithm.

(Nec) seems to require much more semantic filtering based on privileged epistemic position than Mix might need.

Moreover, **(Untyped)** is formally weaker than type-shifting with **(Nec)**. **(Nec)** does not restrict type-shifts to a *kind* of expression. Meanwhile, **(Untyped)** *only* applies to expressions which feature untyped terms. The calculations featuring untyped terms form a proper subclass of the whole space of calculations, unlike the space of terms which may type-shift in (totally general) type-shifting.

Mix also appears straightforwardly simpler and more parsimonious than type-shifting with **(Nec)**, as it has *fewer rules*. Even if Mix requires a strong meta-rule to characterise legitimate calculations, type-shifting does as well, whilst Mix employs no additional type-shifting rules. Finally, Mix does not assign the untyped terms arbitrary types (§ 4.5.5).

4.5 Is Mix Pseudo-Typed?

Here we briefly outline why Mix is conceptually independent of other plausible semantic systems. We consider several dissimilar system[177]. Some candidates are independently implausible, so are excluded. Others have equivalence-invariant properties which Mix does not, e.g. procedural ineffectiveness[178], showing the independence of Mix.

On some level, Mix is equivalent to total typing. Both are descriptively adequate, i.e. they compute and agree on the meanings of some key class of expressions (perhaps all of them). But this *computational* equivalence is a minimum requirement for all therapeutic semantics of natural language. So all plausible formal semantics are computationally equivalent *anyway*.

Instead, we focus on the lack of *strong* equivalences, i.e. structure preserving maps between the semantics and syntax of Mix and that of another candidate system.

4.5.1 Total Untyping

The first system, S_1, is total *un*typing. S_1 has *absolutely* no structure, except for an excessively strong calculation rule:

(Utopian) For any expression, *suitable* semantic calculations are legitimate.

It's doubtful that an equivalence between Mix and S_1 could even be *defined*, as S_1 has no firm structure. Specifically, there is no suitable equivalent (sub-) rule(s) for the rules of Mix. For example, there can be no map between **(Utopian)** and **(A)**, as the former is far less restricted than the latter. Mapping **(A)** to a restriction of **(Utopian)** requires some semantic property in S_1 (like a type) by which to restrict it. But there are no obvious candidate properties.

S_1 is also intensely procedurally ineffective. There is no clear way to algorithmically define 'suitability', consequently **(Utopian)** (and S_1) cannot be axiomatised so as to provide a *decidable algorithm* for determining meanings.

Hence, S_1 and Mix are *non*-isomorphic.

[177] Another candidate asserts that some logical connectives are *rules* (§ 2.1.1). We reject this account as (1) it is disingenuous, these lexemes are natural language words like any other, (2) it separates off some subset of the connectives from their natural language cousins, and (3) it plausible that type ambiguity extends beyond the connectives (§ 1.8).

[178] Suitable versions of Mix are plausibly algorithmic (§ 4.4).

4.5.2 Multityping

S_2 is multityping, where single lexical items have several types simultaneously (§ 1.8.1).

S_2 also suffers from procedural ineffectiveness problems. In an arbitrary calculation, S_2 has many possible type combinations. Further, as the number of types for an expression is indeterminate[179], it again seems as though there cannot be a decidable algorithm to generate the compositional meanings.

Moreover, S_2 may legitimate the 'wrong' expressions. If there is *some* compositional combination amongst the type-set for the subexpressions, multityping apparently judges this expression meaningful. So S_2 and Mix may even generate *different semantics* (S_2 is then descriptively inadequate, and consequently implausible).

Finally, the semantics of S_2 are mysterious. It's unclear how to interpret a lexical entry having *several types simultaneously*. Meanwhile, Mix can give straightforward, plausible definitions for its untyped (and typed[180]) lexemes.

4.5.3 Enlarged Ontology Simple Typing

A third candidate, S_3, is a typed system with an enlarged ontology, such that precisely the untyped terms have the additional type. Such ontologies might include types e, t, and v (for being untyped).

However, § 4.3 demonstrated that being untyped is systematically different to being typed. To accommodate the interaction between the new (un)types and the original types, S_3 must restrict its derivation rule (and possibly its application rule). This rule is unpalatable, as it is inelegant and non-general. Moreover, if e.g. *only the connectives* are type v, then the structure of the denotation domain $D_{v,D}$ is mysterious. Hence, S_3 is implausible, and does not concern us.

4.5.4 Homonym Theory

The first serious contender is S_4, (stipulative or rule-generated) homonym theory.

S_4 and Mix are clearly non-equivalent, as S_4 is committed to *more lexical entries* than Mix, so they have different lexicons. This even suggests that they generate different *languages*. Hence they cannot be equivalent. Homonym theory is somewhat more *radical* than other (therapeutic) accounts of natural language semantics.

Even if one lexical entry in Mix could 'correspond' to several in S_4, they still have *different acquisition properties* (§ 3.2.1). In S_4, speakers must learn

[179] See § 3.2.
[180] § 4

additional homonym generation-rules or somehow determine the meanings of obscure homonyms of ordinary words (e.g. ditransitive verbal-**and**). Mix has no such commitments, and so more easily fits with our understanding of language acquisition, which does not involve learning such rules or encountering such obscure meanings.

Finally, we might think S_4 is simply implausible. S_4 relied on a 'semantic link' to justify its (radical!) 'carving up' of ordinary terms. An initial problem is that this link may be inexpressible (§ 3.2.2). Furthermore, we have at least one counterexample: the $(e,(e,e))$ meaning of **and** is not plausibly connected to its ancestrally-t (e.g. $(t,(t,t))$) meaning. One meaning is collective, the other fusional (§ 4.2.2). This may pull out the motivational rug from under S_4.

4.5.5 Type-Shifting

Perhaps the most important distinction is between Mix and S_5, the regular (event or extensional) type-shifting account.

Despite their differing overt structure, one might suppose that the untyped lexemes could have the same meaning in S_5 as in Mix. Assuming this, their *entire lexicon* would coincide.

However, there are good semantic reasons to think otherwise. Like S_4, S_5 apparently relies on the difficult lexemes having a *uniform* (non-disjunctive) meaning, which applies to expressions of different type *in the same way*, when it is type-shifted. Our counterexample is **and**, whose meaning apparently depends on the *type* of the terms it conjoins (§ 4.2.2). So this uniform-definitions version of S_5 does not generate the same semantics as Mix. It would also be descriptively inadequate!

Moreover, for each element on the ontology, S_5 requires a different **and**-homonym (likewise for other connectives), as they cannot shift between e.g. ancestrally-t and ancestrally-e types. But we think that these are *the very same word*. Hence, the type-shifter's lexicon is artificially inflated, and so cannot be equivalent to Mix for the same reasons as with S_4.

Most significantly, *Mix does not have to assign arbitrary types to lexemes which do not comfortably fit amongst the types* (§ 3.2.3). Even if the two accounts have the same definitions for the connectives, their structures are different. S_5 assigns some 'base' type to (each homonym of) **and**, e.g. $(e,(e,e))$. But any such choice is a groundless affectation. Meanwhile, in Mix, untyped terms do not have this arbitrary semantic-syntactic 'base' property. This suggests that the two accounts are non-equivalent, but perhaps also that S_5 is unsuitable as a semantic theory for natural language.

5 Conclusion

Montague grammars provide computationally effective compositional semantics, and are particularly valued for their independence from semantic filtering. To maintain compositional simplicity, natural language words are best analysed as having different types in different expressions, yielding type ambiguity. We focused on the ambiguity of connectives.

Simple totally typed systems do not adequately explain type ambiguity. Instead, contemporary accounts introduce some type flexibility. Yet the standard flexible theories, homophone theory and type-shifting, were both unsatisfactory. The main issues were the lack of theoretical economy, questionable mathematical motivation, and arbitrary type assignment to the connectives. They may also violate speaker intuitions.

I suggest that the problem is attempting to *totally* type natural language. A mixed system, Mix, allows for some untyped lexemes. These are given independent definitions, and we independently explain how they are used in semantic calculation. Hence, Mix adequately explains the uses of connectives. A rich area of research would be investigating whether Mix explains other type ambiguity, e.g. intensifiers or ambitransitive verbs.

Mix does not seem to suffer from extra issues of procedural effectiveness, nor does it obviously require a privileged epistemic position to effectively calculate. This is potentially *dissimilar* to the alternative accounts, and their implicit meta-rules.

There is good reason to think that Mix is conceptually independent and more theoretically parsimonious than the alternatives. It possibly agrees with controversial speaker intuitions, should they exist. Finally, it does not assign arbitrary types to naturally occurring lexemes which do not fit well in the type hierarchy.

Bibliography

[1] Barendregt H.P. *The Lambda Calculus: Its Syntax and Semantics*, 1984.

[2] Bar-Hillel Y. *Language and Information: Selected Essays on Their Theory and Application*, 1964.

[3] Barwise J., Cooper R. *Generalized Quantifiers and Natural Language*, Ling. and Phil. 4(2): 159–219, 1981.

[4] Boolos G. *The Iterative Conception of Set*, Jour. of Phil. 68(8):215-231, 1971.

[5] Bowler N., Forster T.E. *Internal Automorphisms and Antimorphisms of Models of NF*, 2017.

[6] Buszkowski W. *Mathematical Linguistics and Proof Theory*, Handbook of Logic and Language, van Benthem and ter Meulen (eds.), 1997.

[7] Champollion L. *Compositional Semantics and Event Semantics*, BRIDGES conference notes, 2014.

[8] Champollion L. *Integrating Montague Semantics and Event Semantics*, ESSLLI lecture notes, 2014. `lingbuzz/002143`

[9] Champollion L. *Ten Men and Women Got Married Today: Noun Coordination and the Intersective Theory of Conjunction*, Jour. of Sem. 33(3.1): 561-662, 2016.

[10] Chomsky N. *Three Models for the Description of Language*, 1956.

[11] Chomsky N. *Aspects of the Theory of Syntax*, 1965.

[12] Chomsky N. *Knowledge of Language: Its Nature, Origin, and Use*, 1986.

[13] Collins J. *Naturalism in the Philosophy of Language; or Why There Is No Such Thing as Language*, New Waves in the Philosophy of Language, Sawyer (ed.), 2010.

[14] Cresswell M.J. *Categorical Languages*, 1977.

[15] Davidson D. *The Logical Form of Action Sentences*, The Logic of Decision and Action, Rescher (ed.), 81-94, 1967.

[16] Davidson D. *On Saying That*, Synthese 19(1/2): 130-146, 1968.

[17] Dowty D. *Type Raising, Functional Composition, and Non-Constituent Conjunction*, Oehrle et al. (eds.), 153-197, 1998.

[18] Fine K. *Relatively Unrestricted Quantification*, Absolute Generality, Rayo and Uzquiano (eds.), 2006.

[19] Fodor J.A. *The Modularity of Mind*, 1983.

[20] Frigg R., Hartmann S. *Models in Science*, SEP, Zalta (ed.), 2012.

[21] J. T. F. GAMUT *Logic, Language, and Meaning, vol. 2*, 1991.

[22] Geach P.T. *A Program for Syntax*, Synthese 22(1/2): 3-17, 1970.

[23] Grice P. *Studies in the Way of Words*, 1991.

[24] Hankamer J., Sag I. *Deep and Surface Anaphora*, Ling. Inquiry 7(3): 391-428, 1976.

[25] Heim I., Kratzer A. *Semantics in Generative Grammar*, 1998.

[26] Hendriks H. *Studied Flexibility*, 1993.

[27] von Humboldt W. *On Language*, 1836.

[28] Humberstone L. *Geach's Categorical Grammar*, Ling. and Phil., 28(3): 281-317, 2005.

[29] Jacobson P. *Direct Compositionality and Variable-Free Semantics: The Case of Binding into Heads*, 2001.

[30] Jacobson P. *Direct Compositionality and Variable-Free Semantics: The Case of "Principle B" Effects*, 2003.

[31] Kayne R. *The Antisymmetry of Syntax*, 1994.

[32] Keenan E.L., Faltz L. *Logical Types for Natural Language*, 1978.

[33] Keenan E.L., Faltz L. *Boolean Semantics for Natural Language*, 1985.

[34] Kibort A. *Transitivity*, Grammatical Features, 2008.

[35] Kratzer A. *Situations in Natural Language Semantics*, SEP, Zalta (ed.), 2017.

[36] Kripke S. *Semantical Analysis of Modal Logic*, ZML 9: 67–96, 1963.

[37] Kripke S. *Outline of a Theory of Truth*, Jour. of Phil. 72(19): 690–716, 1975.

[38] Kunen K. *Set Theory*, 2011.

[39] Ladusaw W. *Principles of Semantic Filtering*, 1986.

[40] Lambek J. *Type Grammar Revisited*, Logical Aspects of Computational Linguistics, Lecomte (ed.), 1999.

[41] Lasersohn P.N. *Event-Based Semantics*, Encyclopedia of Language and Linguistics, 2nd ed., Vol. 4, Brown (ed.), 2006.

[42] Lewis D. *General Semantics*, Semantics of Natural Language, Davidson and Harman (eds.), 169-218, 1972.

[43] Liefke K., Hartmann S. *Montague Reduction, Confirmation, and the Syntax-Semantics Relation*, Preprint, 2014.

[44] Link G. *Algebraic Semantics in Language and Philosophy*, 1998.

[45] Magidor O. *The Last Dogma of Type Confusion*, Proc. Arist. Soc., New Series 109: 1-29, 2009.

[46] Al-Mansour N.S. *A Brief History of Arabic Linguistics*, 2001.

[47] Meinong A. *Untersuchungen zur Gegenstandstheorie und Psychologie*, 1904, trans. in Realism and the Background of Phenomenology, Chisholm (ed.), 1981.

[48] Montague R.M. *Universal Grammar*, 1970.

[49] Montague R.M. *English as a Formal Language*, 1970.

[50] Montague R.M. *The Proper Treatment of Quantification in Ordinary English*, 1973.

[51] Moortgat M. *Categorial Grammar and Formal Semantics*, 2002.

[52] Muskens R. *Type-Logical Semantics*, Routledge Encyclopedia of Philosophy Online, Craig (ed.), 2011. `rep.routledge.com/article/U061`

[53] *How Many Words Are There in the English Language?*, OED Blog, 2016. `en.oxforddictionaries.com/explore/does-english-have-most-words`

[54] *and, conj.1, adv., and n.1*, OED Online, 2018. `oed.com/view/Entry/7283?rskey=CwZwPV&result=3`

[55] Partee B. *Compositionality*, Varieties of Formal Semantics, Landman and Veltman (eds.), 1984.

[56] Partee B. *Noun Phrase Interpretation and Type-Shifting Principles*, Studies in Discourse Representation Theory and the Theory of Generalized Quantifiers, Groenendijk et al. (eds.), 115-143, 1987.

[57] Partee B., Rooth M. *Generalized Conjunctions and Type Ambiguity*, 1983.

[58] Pylkkänen L. *Introducing Arguments*, 2008.

[59] Rodriguez-Pereyra G. *Nominalism in Metaphysics*, SEP, Zalta (Ed.), 2015.

[60] Roelofsen F. *Two Alternatives for Disjunction: an Inquisitive Reconciliation*, 2016.

[61] Russell B. *Mathematical Logic as Based on a Theory of Types*, Amer. Jour. of Math. 30(3): 222-262, 1908.

[62] Strawson P.F. *On Referring*, Mind 59(235): 320-344, 1950.

[63] Tarski A. *The Semantic Conception of Truth: and the Foundations of Semantics*, 1944.

[64] Wigner E.P. *The Unreasonable Effectiveness of Mathematics in the Natural Sciences*, Comm. Pure Appl. Math. 13: 1-14, 1960. ‘

[65] Winter Y. *Flexibility Principles in Boolean Semantics*, 2001.

[66] Winter Y., Zwarts J. *Event Semantics and Abstract Categorial Grammar*, The Mathematics of Language, Kanazawa et al. (eds.), 2011.