SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Montague's Intensional Logic for Computational Semantics of Human Language

av

Axel Ljungström

2018 - No K16

Montague's Intensional Logic for Computational Semantics of Human Language

Axel Ljungström

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Roussanka Loukanova

2018

Abstract

This thesis concerns methods of rendering human language expressions into mathematical logic as a means of representing meaning in a computational manner. In particular, the work aims to show how rendering into Montague's intensional logic can circumvent some of the problems with rendering into first-order predicate logic. The first part presents some necessary preliminaries regarding formal grammar in computational linguistics. The second part considers first-order predicate logic for semantic representations of human language and includes comments both on its advantages and its disadvantages for such purposes. In the third and final part, Montague's intensional logic is presented together with a Montagovian grammar AGr. Rules for rendering human language into the logic are introduced and are used to resolve some of the problems with rendering into first-order predicate logic.

Acknowledgements

I would like to thank my supervisor Roussanka Loukanova for all the time and energy she has devoted to helping me with this thesis. I could not have asked for someone more helpful and dedicated. I would also like to thank Erik Palmgren for his valuable feedback.

Contents

1 Introduction

One of the most important ideas to come out of the invention of predicate logic was that semantics of human language (HL) sentences can be treated mathematically, using mathematical logic.

In this essay, I take on the task of investigating some of the initial and fundamental ideas of computational approaches to rendering HL expressions into mathematical logic for semantic representations. I shall present two of the most influential theories of the 20th century — rendering into first order predicate logic (FOL) and rendering into Montague Intensional Logic, see Montague [12]. Both approaches have had an important impact on the development of computational semantics of HL. My goal is to show that FOL can serve as a basis of formal semantics of HL, but that it needs to be radically extended if it is to capture many vital semantic aspects. My intention is to illustrate that a significant achievement in this direction was made, for the first time, by Montague [12]. In the final part of the essay, I shall have a short section on more contemporary approaches to computational semantics of HL.

2 Languages

Before we can say anything about renderings of languages we need to establish what a language is. A language is defined in Definition 2.1–2.4, see Hopcroft, Motwani and Ullman [5].

Definition 2.1. Let $\Sigma = \{s_1, s_2, \ldots, s_n\}$ be a set containing $n \geq 1$ (fixed) different *symbols* s_i. We call Σ an *alphabet*.

Definition 2.2. Let $x_1, x_2, \ldots, x_k \in \Sigma$ for some fixed $k \geq 0$. We then call $x_1 x_2 \ldots x_k$ a string. When $k = 0$ we refer to the empty string. We denote this ϵ.

Notation 1. The set of strings of length $k \geq 0$ formed by the alphabet Σ is denoted Σ^k. We let $\Sigma^0 = \{\epsilon\}$.

Definition 2.3. Let $\Sigma^* = \bigcup_{k \geq 0} \Sigma^k$. We call this the *Kleene closure* of Σ.

Definition 2.4. Let $L \subseteq \Sigma^*$. We then say L is a *language*.

Any human language is a language in this formal sense. English, in its most simple form, is the language consisting of the alphabet $\Sigma = \{a, b, c, \ldots, x, y, z\}$ where the formation rule for each Σ^k is that all the strings in Σ^k are also well-formed English language words.

2.1 Context-Free Grammars

One of the central ideas in computational linguistics is that of a *context-free grammar* (CFG). A CFG is, intuitively, a system of generative rules that define

which strings over a given alphabet are to be generated as strings of the language in question. The rules applied during the generation of a string define its syntactic structure.

Formally, we define a CFG as follows, see Hopcroft, Motwani and Ullman [5]:

Definition 2.5. A context-free grammar is any tuple $G = (V, T, P, S)$, where:

1. V is a finite set of *nonterminals*, also called *syntactic categories* (particularly, when the grammar is related to HL), each of which represents its own language. In English, an example is the syntactic category IV, i.e. the language of intransitive verbs

2. T, the *terminals*, is the alphabet which is used to form the strings in each category of V

3. P is a finite set of *rules*. A rule is on the form $C \to X_1, \ldots, X_n$ where $C \in V$ is the *head* of the rule, $\to$ is the rule symbol and each $X_i \in V \cup T$

4. S is a special nonterminal in V called the *start symbol*. The start symbol can be thought of as denoting the major syntactic category

Definition 2.6. Let $\alpha\beta\gamma \in (V \cup \Sigma)^*$. If there is a rule $\beta \to \delta$ in P, we write $\alpha\beta\gamma \Rightarrow \alpha\delta\gamma$ and call this a *derivation* from $\alpha\beta\gamma$ to $\alpha\delta\gamma$. If B can be derived from A using 0 or more derivations we simply write $A \Rightarrow^* B$, see Aho and Ullman [1].

Definition 2.7. Let $G = (V, T, P, S)$ be a context-free grammar. We say $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$ is a *language over* G.

Example 2.1. Consider the following grammar:

$$G = (\{\mathsf{N},\mathsf{I},\mathsf{S}\}, \{0, 1, \ldots, 9\}, \mathsf{P}, \mathsf{S}) \tag{1}$$

where P contains the following rules:

$$\begin{aligned} \mathsf{S} &\to \mathsf{N} \\ \mathsf{N} &\to \mathsf{NI} \\ \mathsf{N} &\to \epsilon \\ \mathsf{I} &\to 0 \mid 1 \mid 2 \mid \ldots \mid 9 \end{aligned} \tag{2}$$

Now, $L(G)$ will contain any string of natural numbers. Consider for instance this derivation of the string 132:

$$\begin{aligned} \mathsf{S} &\Rightarrow \mathsf{N} && \text{(3a)} \\ &\Rightarrow \mathsf{N}2 && \text{(3b)} \\ &\Rightarrow \mathsf{N}32 && \text{(3c)} \\ &\Rightarrow \mathsf{N}132 && \text{(3d)} \\ &\Rightarrow \epsilon 132 = 132 && \text{(3e)} \end{aligned}$$

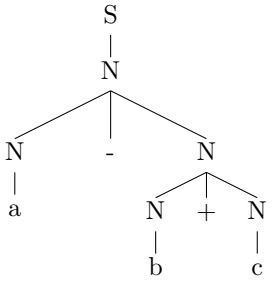
Despite their computation power, CFGs are limited. Consider the following example:

Example 2.2. $G = (\{\mathsf{S},\mathsf{N}\}, \{a, b, c, +, -\}, \mathsf{P}, \mathsf{S})$ where P consists of the following rules:

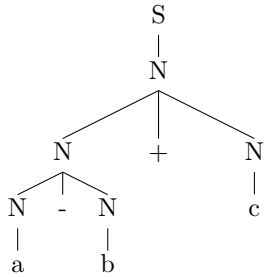
$$\begin{aligned} \mathsf{S} &\to \mathsf{N} \\ \mathsf{N} &\to \mathsf{N} + \mathsf{N} \\ \mathsf{N} &\to \mathsf{N} - \mathsf{N} \\ \mathsf{N} &\to a \mid b \mid c \end{aligned} \tag{4}$$

We now have $a - b + c \in L(G)$. There are two derivations of this. We describe these using the following trees:

(5)



(6)



Cases like (5)–(6) constitute a serious limitation of basic CFGs. $L(G)$ only takes into account *which* expressions can be rendered and not *how* they are rendered. If a computer were to implement a grammar as the above to treat basic arithmetic we would run into problems. Replace, for instance, a by 2, b by 2 and c by 1. In (5), the bottom right subtree represents adding 2 to 1, i.e. 3. This is then subtracted from a. So (5) represents $2 - (2 + 1)$. In the same manner, (6) represents $(2 - 2) + 1$. Consequently, we interpret $2 - (2 + 1)$ in the same way as we interpret $(2 - 2) + 1$ which, obviously, constitutes a severe problem if our grammar is to represent some system of arithmetic. Often, we are not interested only in the fact that a string can be derived but also in which way the string is derived. We say that grammars that give rise to more than one derivation of one string is *ambiguous*. To eliminate these ambiguities we make the following definition:

Definition 2.8. Let G be a CFG. We define the language of derivation trees $\mathsf{DerTree}(G)$ as follows:

1. Every vertex is labelled with some $w \in V \cup T \cup \{\epsilon\}$.
2. The root is labelled S
3. The internal vertices are labelled only with elements V
4. If v is a vertex labelled X, $v_1, \ldots, v_n$ are its daughters, labelled $X_1, \ldots, X_n$ respectively, then $X \to X_1, \ldots, X_n$ is a rule in P
5. If a vertex is labelled ϵ it is
   (a) a leaf
   (b) the only daughter of its parent

See Hopcroft and Ullman [6].

We shall refer to the tree grammar $\mathsf{DerTree}(G)$ of G as a *phrase structure grammar* of G and our goal is, essentially, to construct a grammar G which gives rise to a grammar $\mathsf{DerTree}(G)$ whose elements correspond English language expressions in a way that respects their phrase structure.

2.2 Syntactic Properties of the English Language

We have now covered the necessary preliminaries to state some of the syntactic properties of the baby-version of English we are going to consider in this essay. We do this in the style of a CFG which we endow with a phrase structure.

2.2.1 Syntactic Categories

In Table 1 we introduce a number of syntactic categories which, with simplifications, correspond to established syntactic structures in theoretical and computational linguistics. I want to stress that this is an incredibly simplified version of only a small fraction of English. First, it will treat some phrases such as, for instance, "believe that" and "attempt to" as (compound) words when clearly they are in fact more complex constructions. Furthermore, there are many other linguistic features that CFGs cannot handle in a satisfactory way. For instance, if we were to use a CFG to attack the problems of plural/singular agreement and gender agreement, we would be required to add a large number of new rules and categories which are alien to common linguistic practice. For linguistic details of English syntax, see Sag et al. [16] and Kim and Sells [8]. For a more computational perspective, using methods of mathematical logic, see Loukanova [9–11]. For a given formal grammar G and a HL L, we say that (a) G *undergenerates* L, when grammatical expressions of L are not in the language $G(L)$ generated by G; (b) G *overgenerates* L, when G generates expressions that are not in L. The toy CFG, which we present here, both undergenerates and overgenerates English language, with respect to English syntax. That is,

it both fails to generate many correct grammatical expressions and succeeds in generating many ungrammatical expressions. It is important to point out these details. However, for our purposes they can be ignored. The grammar we are presenting is only intended to serve as a simple introduction of some of the fundamental concepts in the syntax of human language. We shall see that this will be needed especially in Section 4.

Syntactic Category	Description	(Example) Words
S	Sentence	–
NP	Noun Phrase	John, he, she
N	Common Noun	boy, cat
IV	Intransitive Verb	smokes, sings
VP	Verb Phrase	runs away
TV	Transitive Verb	gives
Det	Determiner	the, some, every
P	Preposition	by, on
PP	Preposition Phrase	under the bridge
Adj	Adjective	blue, ugly
Adv	Adverb	rapidly, helplessly
AdjP	Adjective Phrase	silly and blue
SAdv	Sentence Adverb	possibly
SCP	Sentence-Complement Verb	believe that, wish that
ICP	Infinitive-Complement Verb	try to, attempt to
Conj	Coordinators	and, or
Neg	Negation	does not

Table 1: Toy Context Free Grammar

S, the sentence category, is to play the role of our *initial symbol*, by standard terminology in formal grammars.

Let us, for the rest of the essay, restrict ourselves to the following set Γ of basic words, as a lexicon, partitioned into sets of words of syntactic categories that are *parts of speech* (POS). I.e., the lexicon Γ is categorised by POS categories: common nouns (N), intransitive verbs (IV), transitive verbs (TV) determiners (Det), prepositions (P), (premodifying) adjectives (Adj), adverbs (Adv), sentence adverbs (SAdv), sentence-compliment verbs (SCP), infinitive-complement verbs (ICP), coordinators (Conj) and negations (Neg). For detailed, grammatical information about POS and syntactic categories see Huddleston and Pullum [7].

$$\begin{aligned}\Gamma =& L_{\mathrm{N}} \cup L_{\mathrm{N_{NP}}} \cup L_{\mathrm{IV}} \cup L_{\mathrm{TV}} \cup L_{\mathrm{Det}} \cup L_{\mathrm{P}} \cup L_{\mathrm{Adj}} \cup L_{\mathrm{Adv}} \cup L_{\mathrm{SAdv}} \cup \\ & L_{\mathrm{SCP}} \cup L_{\mathrm{ICP}} \cup L_{\mathrm{Conj}} \cup L_{\mathrm{Neg}}\end{aligned} \tag{7}$$

where L_{N} is the set of the words, of POS noun (N), generated by the rule (8a); $L_{\mathrm{N_{NP}}}$ is the set of the proper names generated by the rule (8b); etc.

$$\mathrm{N} \to \mathrm{cat} \mid \mathrm{boy} \mid \ldots \tag{8a}$$

$$N_{NP} \rightarrow \text{Serge} \mid \text{Jacques} \mid \text{he} \mid \text{she} \mid \text{he} \mid \text{it} \mid \ldots \tag{8b}$$
$$IV \rightarrow \text{sing(s)} \mid \text{smoke(s)} \mid \text{will sing} \mid \text{will smoke} \mid \text{sang} \mid \text{smoked} \mid \ldots \tag{8c}$$
$$TV \rightarrow \text{is taller than} \mid \text{writes} \mid \ldots \tag{8d}$$
$$P \rightarrow \text{by} \mid \text{under} \mid \ldots \tag{8e}$$
$$Conj \rightarrow \text{and} \mid \text{or} \mid \ldots \tag{8f}$$
$$Det \rightarrow \text{some} \mid \text{every} \mid \text{the} \mid \ldots \tag{8g}$$
$$Adv \rightarrow \text{well} \mid \text{rapidly} \mid \ldots \tag{8h}$$
$$Adj \rightarrow \text{silly} \mid \text{blue} \mid \ldots \tag{8i}$$
$$SAdv \rightarrow \text{necessarily} \mid \text{possibly} \mid \ldots \tag{8j}$$
$$SCP \rightarrow \text{thinks that} \mid \ldots \tag{8k}$$
$$ICP \rightarrow \text{tries to} \mid \ldots \tag{8l}$$
$$Neg \rightarrow \text{does not} \mid \ldots \tag{8m}$$

2.2.2 Context Free Rules

We now introduce a set of context free rules, called the *phrase structure rules* presented as follows in (9a)–(9t). Together, (8a)–(8m) and (9a)–(9t), are the rules of a CFG, which we use in this thesis to generate a small fragment of English.

Phrase Structure Rules

$$S \rightarrow NP\ VP \tag{9a}$$
$$S \rightarrow SCP\ S \tag{9b}$$
$$S \rightarrow SAdv\ S \tag{9c}$$
$$S \rightarrow S\ Conj\text{-}S \tag{9d}$$
$$Conj\text{-}S \rightarrow Conj\ S \tag{9e}$$
$$NP \rightarrow N_{NP} \tag{9f}$$
$$NP \rightarrow Det\ N \tag{9g}$$
$$NP \rightarrow NP\ Conj\text{-}NP \tag{9h}$$
$$Conj\text{-}NP \rightarrow Conj\ NP \tag{9i}$$
$$VP \rightarrow IV \tag{9j}$$
$$VP \rightarrow IV\ PP \tag{9k}$$
$$VP \rightarrow IV\ Adv \tag{9l}$$
$$VP \rightarrow TV\ NP \tag{9m}$$
$$VP \rightarrow ICP\ VP \tag{9n}$$
$$VP \rightarrow Neg\ VP \tag{9o}$$
$$VP \rightarrow VP\ Conj\text{-}VP \tag{9p}$$
$$Conj\text{-}VP \rightarrow Conj\ VP \tag{9q}$$

$$\text{PP} \rightarrow \text{P NP} \tag{9r}$$
$$\text{Adj} \rightarrow \text{Neg Adj} \tag{9s}$$
$$\text{AdjP} \rightarrow \text{AdjP Conj-AdjP} \tag{9t}$$
$$\text{Conj-AdjP} \rightarrow \text{Conj AdjP} \tag{9u}$$

Note that the words in the set Γ, in (7), are the terminal symbols of our CFG, or simply, words [1] for our purposes here. The rules (8a)–(8m) are called its *terminal rules*, in terminology of formal grammars, or lexical rules, in the terminology of computational and formal grammars of HL. The rules (9a)–(9u) are called *phrasal rules*.

I'm stressing here that the rules above are only to be regarded as an *introductory idea* of how developing a computational grammar of HL can be approached. Our CF rules generate expressions of HL and , simultaneously, assign syntactic categories to the generated expressions. Hence, the derivations by our CF rules endow the generated expressions with an internal syntactic structure. In addition, CFGs are a good source of algorithms for parsing expressions into tree-structure analyses. This is a well-established standard approach in foundations of parsers in computer science and computational linguistics, see Hopcroft and Ullman [6].

Subcategorisation Here, I explain briefly an important technique for removing some overgeneration for CFGs. This was was among the first realisations in computational grammar of human language. The rules (8a), (8b) and (9f) showcase this technique.

If we were to place the proper nouns and common nouns together in a set $L_{\text{N}} = \{\text{Serge}, \text{Jacques}, \text{cat}, \text{boy}, \dots\}$ of POS N (which would be generated by the rule $\text{N} \rightarrow \text{Serge}, \text{Jacques}, \text{cat}, \text{boy}, \dots$), then our CFG would generate ungrammatical expressions such as "The Serge sings".

The technique that we use to deal with this problem is called *subcategorisation*. What this means is that syntactic categories, which share some common feature, are split into subcategories. This is what happens in our CFG where we have the following division of POS N:

1. N, in (8a), is our syntactic POS category of words that may be incomplete NP

2. N_{NP}, in (8b), is the syntactic POS category of words that are NPs, whilst also of nominal POS

Our CFG does, however, still suffer from some overgeneration — it lacks mechanisms to handle grammatical agreement between the major verb of a sentence, i.e., the *grammatical head verb* in a sentence, and its subject NP, as

[1] In computational linguistics, the notion of a word is a more complex.

shown in (10)–(12):

$$
\begin{aligned}
& & \mathrm{S} &\Rightarrow \mathrm{NP\ VP} \\
& & &\Rightarrow \mathrm{NP\ Conj\text{-}NP\ VP} \\
& & &\Rightarrow \mathrm{NP\ Conj\ NP\ VP} \\
& \text{* overgeneration *} & &\Rightarrow \mathrm{NP\ Conj\ NP\ IV} \\
& & &\Rightarrow \mathrm{N_{NP}\ Conj\ NP\ IV} \\
& & &\Rightarrow \mathrm{N_{NP}\ Conj\ Det\ N\ IV} \\
& & &\Rightarrow \cdots \Rightarrow \text{Serge and the cat sings}
\end{aligned}
\tag{10}
$$

$$
\begin{aligned}
& & \mathrm{S} &\Rightarrow \mathrm{NP\ VP} \\
& & &\Rightarrow \mathrm{NP\ IV} \\
& \text{* overgeneration *} & &\Rightarrow \mathrm{N_{NP}\ IV} \\
& & &\Rightarrow \cdots \Rightarrow \text{Serge sing}
\end{aligned}
\tag{11}
$$

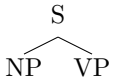
$$
\begin{aligned}
& & \mathrm{S} &\Rightarrow \mathrm{NP\ VP} \\
& & &\Rightarrow \mathrm{NP\ VP\ Conj\text{-}VP} \\
& & &\Rightarrow \mathrm{NP\ VP\ Conj\ VP} \\
& \text{* overgeneration *} & &\Rightarrow \mathrm{NP\ IV\ Conj\ VP} \\
& & &\Rightarrow \mathrm{NP\ IV\ Conj\ IV} \\
& & &\Rightarrow \mathrm{N_{NP}\ IV\ Conj\ IV} \\
& & &\Rightarrow \cdots \Rightarrow \text{Serge sing and smokes}
\end{aligned}
\tag{12}
$$

Concerning such problems of overgeneration, recall the explanation in Section 2.2.1.

For more detailed and developed grammars similar to ours that intend, in a computation way, to treat these problems, see Sag et al. [16] and Kim and Sells [8] (linguistic explanations) and Loukanova [10, 11] (using mathematical logic). Here, we leave the semantics of plurals for future work.

2.2.3 Tree Structures

We note that the above rules make it possible to analyse the generation of sentences by trees, called *phrase structures*, alternatively, *tree structures*, which correspond to the CF rules. For instance, (9a) can be represented by (13):



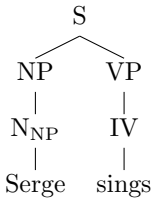
(13)

If we are not interested in the internal structure of a certain subtree we shall replace it by a triangle.

Example 2.3. We give the phrase structure for the following sentences:

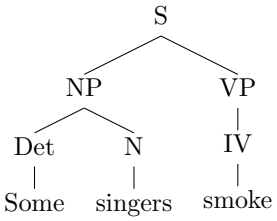
(a) Serge sings

(14)



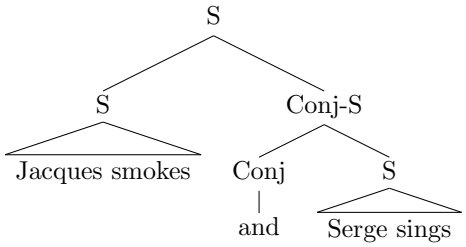
(b) Some singers smoke

(15)



(c) Jacques smokes and Serge sings

(16)



2.3 Semantic Properties of English

Before we attempt to integrate the semantics of English with the syntax of English we must make clear what we mean by "semantics". Roughly speaking, semantics is the study of meaning. It is an integral part of any language. Let us consider an example from mathematics. When we write $e^{i\pi} = -1$ we immediately see from the syntax of our mathematical system that, unlike $e^{=i}\pi = (-)1$, this is a well-formed expression. It is, however, not the fact that $e^{i\pi} = -1$ is well-formed that makes it interesting. Clearly, it carries with it some other information. Said in a very simplified way, it is this information that we call the meaning of the expression. The exact same thing applies to HL. Formally, the English language sentence "Serge loves to smoke" is just a collection of strings from some subset of some Kleene closure based on some alphabet Σ. Still, it is clear to us that this sentence tells us something more - it tells us that Serge loves to smoke. Therefore, we say that the meaning of "Serge loves to smoke" is the condition which needs to be satisfied for the sentence to be true.[2]

We make the following assumptions:

[2]This truth-conditional framework is largely due to Tarski [17] and Davidson [2]

An1 The semantic value of a sentence is a truth value (i.e. 1 or 0). The meaning of a sentence is its truth-conditions

An2 The semantic value of a sentence is determined by the semantic values of its components and the way its components are composed. This is known as *Frege's principle* or *the principle of compositionality*

An3 The semantic values of *modal expressions*, e.g., sentences that include adverbs such as "necessarily" and "possibly", are evaluated with respect to a set of *possible worlds*. So, informally, "Necessarily, it is raining" is true iff it is raining in all possible worlds

The above assumptions are necessary for our project to succeed. It should be noted that An1, An2 and, especially, An3 have been criticised, primarily on philosophical, but also on linguistic and computational grounds. Nevertheless, they have been standard starting points in mathematical foundations of computational semantics. Since this is an essay in mathematics and not in philosophy, I shall take them as valuable starting point.

3 First-Order Logic

In this part of the essay we consider rendering English into FOL. This approach, we shall see, will not take us all the way but it lays the foundation of any more advanced approach. In particular, it is fundamental to understand how renderings into FOL work and in what way they are limited to be able to appreciate the theories presented in Section 4.

3.1 Syntax

Here we present the formal syntax for an initial choice of a language of FOL, L_1. We start by giving a traditional definition, by induction:

Syntax of L_1

1. The formal syntax of L_1 syntactic categories Const_{L_1}, $\mathsf{PredSymb}_{L_1}$, Vars and $\mathsf{Formulae}_{L_1}$

   (i) $\mathsf{Const}_{L_1} :\equiv \{c_0, c_1, \ldots, c_n\}$, for a fixed $n \in \mathbb{N}$, is the set of individual constants, which we also call (individual) names

   (ii) $\mathsf{PredSymb}_{L_1} :\equiv \{P_1^{i_1}, \ldots, P_m^{i_m}\}$, for a fixed $m \in \mathbb{N}$, is the set of predicate symbols, i.e., predicate constants, where, for each $j = 1, \ldots, m$, $P_j^{i_j}$ is a predicate symbol of arity $i_j \in \mathbb{N}$

   (iii) $\mathsf{Vars} :\equiv \{x_0, x_1, x_2, \ldots\}$ is a countable set of symbols, called the variables of L_1

We also require:

$$\mathsf{Const}_{L_1} \cap \mathsf{PredSymb}_{L_1} = \mathsf{Const}_{L_1} \cap \mathsf{Vars} = \mathsf{PredSymb}_{L_1} \cap \mathsf{Vars} = \varnothing \quad (17)$$

We refer to the objects of categories (i)–(ii) as *non-logical constants* and to the objects in categories (i) and (iii) as *individual terms*:

$$\mathsf{Terms}_{L_1} :\equiv \mathsf{Const}_{L_1} \cup \mathsf{Vars} \quad (18)$$

2. $\mathsf{Formulae}_{L_1}$, the set of formulae of L_1, is defined recursively, as follows:

   Definition 3.1 (Formulae of L_1)**.**

   (i) If P is a predicate symbol of arity n, i.e., $P \in \mathsf{PredSymb}_n$, and $t_1, \ldots, t_n$ are individual terms, i.e., $t_1, \ldots, t_n \in \mathsf{Terms}_{L_1}$ (which are not necessarily distinct), then $P(t_1, \ldots, t_n)$ is a formula, called *atomic formula*.

   (ii) If φ is a formula, then $\neg\varphi$ is a formula

   (iii) If φ and ψ are formulas, then $(\varphi \wedge \psi)$ is a formula

   (iv) If φ and ψ are formulas, then $(\varphi \vee \psi)$ is a formula

   (v) If φ and ψ are formulas, then $(\varphi \rightarrow \psi)$ is a formula

   (vi) If φ and ψ are formulas, then $(\varphi \leftrightarrow \psi)$ is a formula

   (vii) If φ is a formula and x is a variable, then $\forall x\, \varphi$ is a formula

   (viii) If φ is a formula and x is a variable, then $\exists x\, \varphi$ is a formula.

   (ix) If t_1 and t_2 are idividual terms, i.e. $t_1, t_2 \in \mathsf{Terms}_{L_1}$, then $(t_1 = t_2)$ is a formula

Note that I will very often use metavariables, such as φ and ψ in Definition 3.1 (i)–(viii) above, for the objects of the language in question. For instance, I will often use P or bold-face words such as **sing**, **smoke**, etc., instead of P^i_j to represent predicate symbols and x, y and z to represent variables. Also note the following notational agreement

Notation 2. I will often use the typical $=$ as equivalence depending on the context of discussion. Often, when a and b are expressions of some language L, I will use $a = b$ to say that a is on the form of b.

From now on, I will define large parts of the syntax of a given language by using *Backus Normal Form* BNF-style of recursive definitions.

Notation 3. The symbol $:=$ is used in:

1. variable assignments
2. special recursion terms

The symbol $:\equiv$ is used for definitional introductions, definitions in BNF style, and in the syntactic operator of formal substitution (replacement).

Notation 4. For any language L, the set $\mathsf{Formulae}_L$ of the formulae of L depends on the choice of the sets Const_L and $\mathsf{PredSymb}_L$. We omit the subscript and write $\mathsf{Formulae}$, Const and $\mathsf{PredSymb}$, by assuming a given language L.

Definition 3.2. The set of predicate symbols of arity $i \in \mathbb{N}$ is:

$$\mathsf{PredSymb}_i :\equiv \{P_1^i, \ldots, P_{m_i}^i\}, \quad \text{for } i, m_i \in \mathbb{N} \tag{19}$$

Of course then, by Definition 3.2 we have:

$$\mathsf{PredSymb} = \bigcup_{i \geq 1} \mathsf{PredSymb}_i \tag{20}$$

Formulae of L_1 (BNF)-Style. Definition 3.1 (ii)–(viii) of the set $\mathsf{Formulae}_{L_1}$ of the formulae of L_1 is given in BNF-style by Definition 3.3, (21a)–(21c).

Definition 3.3 (Formulae of L_1 in BNF-Style)**.** Given that $t_1, \ldots, t_n \in \mathsf{Terms}$, $P^i \in \mathsf{PredSymb}_i$, the set of the formulae $\mathsf{Formulae}_{L_1}$ is defined by the recursive rules (21a)–(21c):

$$\Phi :\equiv P^i(t_1, \ldots, t_i) \mid (t_i = t_j) \mid \tag{21a}$$

$$\neg\varphi \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\varphi \to \psi) \mid (\varphi \leftrightarrow \psi) \mid \tag{21b}$$

$$\exists x\varphi \mid \forall x\varphi \tag{21c}$$

Notation 5. Sometimes, we omit parentheses when there is no risk for confusion.

Definition 3.4. For every formula $\varphi \in \mathsf{Formulae}$, we define the set $\mathsf{FreeV}(\varphi)$ of the free variables of φ and the set $\mathsf{BoundV}(\varphi)$ of the bound variables of φ by structural induction over the Definition 3.3 of the formulae of L_1:

1. If $c \in \mathsf{Const}$, then $\mathsf{FreeV}(c) = \mathsf{BoundV}(c) = \varnothing$
2. If $x \in \mathsf{Vars}$, then $\mathsf{FreeV}(x) = \{x\}$ and $\mathsf{BoundV}(x) = \varnothing$
3. If $\varphi = P(t_1, \ldots, t_n)$, where $P \in \mathsf{PredSymb}_n$ and $t_1, \ldots, t_n \in \mathsf{Terms}$, then $\mathsf{FreeV}(\varphi) = \bigcup_{i=1}^n \mathsf{FreeV}(t_i)$, and $\mathsf{BoundV}(\varphi) = \varnothing$
4. If $\varphi = \neg\psi$, where $\psi \in \mathsf{Formulae}$, then $\mathsf{FreeV}(\varphi) = \mathsf{FreeV}(\psi)$ and $\mathsf{BoundV}(\varphi) = \mathsf{BoundV}(\psi)$
5. If $\varphi = (\psi \ * \ \xi)$, where $\psi, \xi \in \mathsf{Formulae}$ and $*$ is either $\wedge$, $\vee$, $\to$, $\leftrightarrow$, or $=$, then $\mathsf{FreeV}(\varphi) = \mathsf{FreeV}(\psi) \cup \mathsf{FreeV}(\xi)$ and $\mathsf{BoundV}(\varphi) = \mathsf{BoundV}(\psi) \cup \mathsf{BoundV}(\xi) - \mathsf{FreeV}(\psi) \cup \mathsf{FreeV}(\xi)$

6. If $\varphi = Qx\,\psi$ where $\psi \in \mathsf{Formulae}$, $x \in \mathsf{Vars}$, and Q is either $\forall$ or $\exists$, then $\mathsf{FreeV}(\varphi) = \mathsf{FreeV}(\psi) - \{x\}$ and $\mathsf{BoundV}(\varphi) = \mathsf{BoundV}(\psi) \cup \{x\}$

   We say that the formula ψ is in the scope of the quantifier Qx in $Qx\,\psi$, and that, the quantifier Qx binds all free occurrences of x in ψ

Definition 3.5. If $\varphi \in \mathsf{Formulae}$ we define the set of variables of φ as:

$$\mathsf{Var}(\varphi) := \mathsf{FreeV}(\varphi) \cup \mathsf{BoundV}(\varphi) \tag{22}$$

Definition 3.6. For each occurrence of $Qx\,\psi \in \mathsf{Formulae}$ in a formula $\varphi \in \mathsf{Formulae}$, where $x \in \mathsf{Vars}$, and Q is either $\forall$ or $\exists$, the formula ψ is the scope of the quantifier Qx in that occurrence. In $Qx\,\psi$, all free occurrences of all variables $y \in \mathsf{FreeV}(\psi)$ in ψ are in the scope of Qx.

Definition 3.7. If $\varphi \in \mathsf{Formulae}$ we call φ a *sentence* iff $\mathsf{FreeV}(\varphi) = \varnothing$.

Therefore, a formula φ is a sentence exactly when, for each variable x, each of the occurrences of x in φ is within the scope of either $\forall x$ or $\exists x$.

3.2 Semantics

To give an account of the semantics of L_1 we first need to introduce some fundamental concepts of semantics of a formal language L. We note that these definitions mainly apply to L_1 as this is the only language we have defined so far but that they can be extended for any formal language L.

Definition 3.8. A *model* $\mathcal{M}$ for a language L is tuplet containing a domain D and an interpretation function I that assigns values to the non-logical constants of L with respect to D. In particular, for L_1 a model $\mathcal{M} = (D, I)$ is an ordered pair where D is a non-empty set and I is a function assigning values to the non-logical constants of L, so that, for the constants of L_1:

$$I(c) \in D, \qquad \text{if } c \in \mathsf{Const} \tag{23a}$$

$$I(P) \subseteq D^n, \quad \text{if } P \in \mathsf{PredSymb}_n \tag{23b}$$

The function I called the *interpretation function* of the model $\mathcal{M}$ for the language L.

Definition 3.9. Let Vars_L denote the set of variables in a language L and let $\mathcal{M} = (D, I)$ be a model for L. Any function $g\colon \mathsf{Vars}_L \to D$ is called a *variable assignment* for L in $\mathcal{M}$, or simply, *assignment* or *valuation* of L.

For L_1, the elements of Const are interpreted as elements of D, by (23a), and the elements of $\mathsf{PredSymb}$ are interpreted as a set of tuplets (with length corresponding to the arity of the constant in question) of some elements of D, by (23b). If we for instance let $D = \{1, 2, 3\}$ and we wish to interpret the binary predicate symbol P as representing the binary predicate '... is greater than ...' we let:

$$I(P) = \{(2, 1), (3, 1), (3, 2)\} \tag{24}$$

since both 2 and 3 are greater than 1 and 3 is greater than 2. The assignment can be seen as a temporary interpretation. Let $I(P)$ be as in (24), let $x, y \in \mathsf{Vars}$ and consider:

$$P(x, y) \tag{25}$$

We have no way of evaluating (25). We need to assign to x and y some elements of the domain, like the interpretation function did for a and b. So if h is an assignment and, for instance, $h(x) = 2$, $h(y) = 1$ with $a, b \in \mathsf{Const}$, (25) can be thought of as (informally!):

$$\text{“}P(2, 1)\text{”} \tag{26}$$

by which we mean that 2 is greater than 1.

Before we move on to give semantic values to the expressions of L_1 I want to make a short clarification on how a model $\mathcal{M} = (D, I)$ relates to human language and, especially, our fragment of English Γ. When we use language we have an idea of what a significant part of the words we use refer to — fundamentally, some refer to objects in the world and some refer to relations between these objects. Ideally, the domain D represents all of the objects we have in mind when we talk and the interpretation function I represents the way in which we connect these objects with the words we use. Hence, I appears to capture a good portion of the meaning of a word. Of course, capturing *all* of linguistics in a model $\mathcal{M} = (D, I)$ seems like an overambitious project. For our fragment Γ this approach does, however, appear to give a fairly adequate picture of how this small part of language works. The power of a model-theoretic approach is that it reflects compositionality very well. We are allowed to set up our domain D in any way we like and we are free to choose the interpretation function I so that it gives words precisely the extension we wish. What happens to the semantic values when we add these expressions together will then be determined by our model in a clear and compositional way. Note that the model-theoretic approach by no means is restricted to L_1 and will be used for the semantics of any language in this essay.

We now move on to give semantic values to the expressions of L_1. The semantic value of a formula φ is always assigned relative to a model $\mathcal{M} = (D, I)$, and an assignment h. This is denoted by $[\![\varphi]\!]^{\mathcal{M},h}$.

Semantics of L_1 The semantics of the language L_1 is given by Definition 3.10. That is, semantic values of the L_1 expressions are defined by structural induction on the syntax of L_1.

Definition 3.10 (Semantics of L_1 Expressions)**.** We assign semantic values to the formulas by using a set $\{0, 1\}$, where 0 corresponds to *false* and 1 corresponds to *true*.

For every model $\mathcal{M} = (D, I)$ and every variable assignment h for L_1 in $\mathcal{M} = (D, I)$, and for every $\alpha \in \mathsf{Formulae} \cup \mathsf{Terms}$, we define the semantic value $[\![\alpha]\!]^{\mathcal{M},h}$ as follows, by structural induction on α:

1. If $\alpha \in \mathsf{Const} \cup \mathsf{PredSymb}$, then $[\![\alpha]\!]^{\mathcal{M},h} = I(\alpha)$
2. If $\alpha \in \mathsf{Vars}$, then $[\![\alpha]\!]^{\mathcal{M},h} = h(\alpha)$
3. If α is a predicate symbol of arity n, i.e., $\alpha \in \mathsf{PredSymb}_n$, and $t_1, \ldots, t_n$ are individual terms, i.e., $t_1, \ldots, t_n \in \mathsf{Terms}_{L_1}$, then $[\![\alpha(t_1, \ldots, t_n)]\!]^{\mathcal{M},h} = 1$ iff $([\![t_1]\!]^{\mathcal{M},h}, \ldots, [\![t_n]\!]^{\mathcal{M},h}) \in [\![\alpha]\!]^{\mathcal{M},h}$
4. If $\alpha \in \mathsf{Formulae}$, then $[\![\neg\alpha]\!]^{\mathcal{M},h} = 1$ iff $[\![\alpha]\!]^{\mathcal{M},h} = 0$. Otherwise, $[\![\neg\alpha]\!]^{\mathcal{M},h} = 0$
5. If $\alpha = (\varphi \wedge \psi)$ with $\varphi, \psi \in \mathsf{Formulae}$, then $[\![\alpha]\!]^{\mathcal{M},h} = [\![(\varphi \wedge \psi)]\!]^{\mathcal{M},h} = 1$ iff $[\![\varphi]\!]^{\mathcal{M},h} = 1$ and $[\![\psi]\!]^{\mathcal{M},h} = 1$. Otherwise, $[\![\alpha]\!]^{\mathcal{M},h} = 0$
6. If $\alpha = (\varphi \vee \psi)$ with $\varphi, \psi \in \mathsf{Formulae}$, then $[\![\alpha]\!]^{\mathcal{M},h} = [\![(\varphi \vee \psi)]\!]^{\mathcal{M},h} = 1$ iff $[\![\varphi]\!]^{\mathcal{M},h} = 1$ or $[\![\psi]\!]^{\mathcal{M},h} = 1$. Otherwise, $[\![\alpha]\!]^{\mathcal{M},h} = 0$
7. If $\alpha = (\varphi \to \psi)$ with $\varphi, \psi \in \mathsf{Formulae}$, then $[\![\alpha]\!]^{\mathcal{M},h} = [\![(\varphi \to \psi)]\!]^{\mathcal{M},h} = 1$ iff $[\![\varphi]\!]^{\mathcal{M},h} = 0$ or $[\![\varphi]\!]^{\mathcal{M},h} = [\![\psi]\!]^{\mathcal{M},h} = 1$. Otherwise, $[\![\alpha]\!]^{\mathcal{M},h} = 0$
8. If $\alpha = (\varphi \leftrightarrow \psi)$ with $\varphi, \psi \in \mathsf{Formulae}$, then $[\![\alpha]\!]^{\mathcal{M},h} = [\![(\varphi \leftrightarrow \psi)]\!]^{\mathcal{M},h} = 1$ iff $[\![\varphi]\!]^{\mathcal{M},h} = [\![\psi]\!]^{\mathcal{M},h} = 1$ or $[\![\varphi]\!]^{\mathcal{M},h} = [\![\psi]\!]^{\mathcal{M},h} = 0$. Otherwise, $[\![\alpha]\!]^{\mathcal{M},h} = 0$
9. If $\alpha = \forall x\, \varphi$ with $\varphi \in \mathsf{Formulae}$ and $x \in \mathsf{Vars}$, then $[\![\alpha]\!]^{\mathcal{M},h} = [\![\forall x\, \varphi]\!]^{\mathcal{M},h} = 1$ iff $[\![\varphi]\!]^{\mathcal{M},h'} = 1$, for all assignments h' such that, for all $v \in \mathsf{Vars} - \{x\}$, $h(v) = h'(v)$
10. If $\alpha = \forall x\, \varphi$ with $\varphi \in \mathsf{Formulae}$ and $x \in \mathsf{Vars}$, then $[\![\alpha]\!]^{\mathcal{M},h} = [\![\forall x\, \varphi]\!]^{\mathcal{M},h} = 1$ iff $[\![\varphi]\!]^{\mathcal{M},h'} = 1$, for some assignment h' such that, for all $v \in \mathsf{Vars} - \{x\}$, $h(v) = h'(v)$
11. If α is $(\beta = \gamma)$ with $\beta, \gamma \in \mathsf{Terms}$, then $[\![(\beta = \gamma)]\!]^{\mathcal{M},h} = 1$ iff $[\![\beta]\!]^{\mathcal{M},h} = [\![\gamma]\!]^{\mathcal{M},h}$.

Lemma 3.1. If φ is a sentence, then for all assignments h, h' we have $[\![\varphi]\!]^{\mathcal{M},h} = [\![\varphi]\!]^{\mathcal{M},h'}$.

Proof. Assume $[\![\varphi]\!]^{\mathcal{M},h} \neq [\![\varphi]\!]^{\mathcal{M},h'}$. Then, for at least one $x \in \mathsf{FreeV}(\varphi)$ we have $h(x) \neq h'(x)$. But since φ is a sentence, we have that $\mathsf{FreeV}(\varphi) = \varnothing$. Consequently, $h(x) = h'(x)$ for all $x \in \mathsf{FreeV}(\varphi)$. This is a contradiction and so the lemma follows. □

By Lemma 3.1 Definition 3.11 makes sense:

Definition 3.11. If φ is a sentence we define $[\![\varphi]\!]^{\mathcal{M}} := [\![\varphi]\!]^{\mathcal{M},h}$, where h is an arbitrary assignment.

Definition 3.12 (Semantic and Logical Consequences)**.**

1. We say that a sentence ψ is a semantic consequence of a set of sentences Γ in $\mathcal{M}$, and write $\Gamma \models_{\mathcal{M}} \psi$, iff:

$$\text{for all } \varphi \in \Gamma,\ [\![\varphi]\!]^{\mathcal{M}} = 1 \implies [\![\psi]\!]^{\mathcal{M}} = 1 \tag{27}$$

2. We say that a sentence ψ is a logical consequence of a set of sentences Γ, and write $\Gamma \models \psi$, iff, for all $\mathcal{M}$:

$$\text{for all } \varphi \in \Gamma,\ [\![\varphi]\!]^{\mathcal{M}} = 1 \implies [\![\psi]\!]^{\mathcal{M}} = 1 \tag{28}$$

Definition 3.13. We say that a sentence φ is a logical truth if $[\![\varphi]\!]^{\mathcal{M}} = 1$ for all models $\mathcal{M}$. We denote this "$\models \varphi$".

Notation 6. If $\Gamma \models \psi$ and $\Gamma = \{\, \varphi \,\}$ we sometimes omit the brackets and write:

$$\varphi \models \psi \tag{29}$$

Example 3.1. Let $\mathcal{M} = (D, I)$ be a model where $D = \{Edith,\ Serge,\ Jacques\}$ and I is defined:

- $I(a) = Edith$
- $I(b) = Serge$
- $I(c) = Jacques$
- $I(P) = \{Edith,\ Serge,\ Jacques\}$
- $I(Q) = \{Edith\}$
- $I(R) = \{(Serge, Edith), (Jacques, Edith), (Jacques, Serge)\}$

For intuition, think of P being interpreted as the unary predicate '... smokes', Q as the unary predicate '... is a woman' and R as the binary predicate '... is taller than ...' so that for instance $R(b, a)$ can be thought of as representing "Serge is taller than Edith". Let us now show, from first principles, that (30) and (31) are correct:

$$[\![P(a) \wedge Q(a)]\!]^{\mathcal{M}} = 1 \tag{30}$$
$$[\![Q(a) \to \exists x R(x, c)]\!]^{\mathcal{M}} = 0 \tag{31}$$

Solution: We start with (30):

$$[\![P(a) \wedge Q(a)]\!]^{\mathcal{M}} = 1 \iff [\![P(a)]\!]^{\mathcal{M}} = 1 \text{ and } [\![Q(a)]\!]^{\mathcal{M}} = 1 \tag{32a}$$
$$\iff [\![P]\!]^{\mathcal{M}}([\![a]\!]^{\mathcal{M}}) = 1 \text{ and } [\![Q]\!]^{\mathcal{M}}([\![a]\!]^{\mathcal{M}}) = 1 \tag{32b}$$
$$\iff Edith \in I(P) \text{ and } Edith \in I(Q) \tag{32c}$$

Since the last statement holds, it must be the case that $[\![P(a) \wedge Q(a)]\!]^{\mathcal{M}} = 1$

Let us now show (31):

$$[\![Q(a) \to \exists x R(x, c)]\!]^{\mathcal{M}} = 1 \iff [\![Q(a)]\!]^{\mathcal{M}} = 0 \text{ or } [\![\exists x R(x, c)]\!]^{\mathcal{M}} = 1 \tag{33a}$$
$$\iff [\![Q]\!]^{\mathcal{M}}([\![a]\!]^{\mathcal{M}}) = 0 \text{ or there is an } h \tag{33b}$$
$$\text{such that } [\![R(x, c)]\!]^{\mathcal{M},h} = 1 \tag{33c}$$

We know that $[\![Q]\!]^{\mathcal{M}}([\![a]\!]^{\mathcal{M}}) \neq 0$. Now, assume there is a h such that $[\![R(x,c)]\!]^{\mathcal{M},h} = 1$. We have:

$$[\![R(x,c)]\!]^{\mathcal{M},h} = 1 \iff (h(x), I(c)) \in I(R) \tag{34a}$$

$$\iff (h(x), Jacques) \in I(R) \tag{34b}$$

Clearly, for none of the three possible values of $h(x)$ the last statement is true. So there can be no h such that $[\![R(x,c)]\!]^{\mathcal{M},h} = 1$. So $[\![\exists x R(x,c)]\!]^{\mathcal{M}} \neq 1$. Finally we conclude that $[\![Qa \to \exists x R(x,c)]\!]^{\mathcal{M}} = 0$.

3.3 Substitution

One of the most important features of FOL is *substitution*. By substitution we mean, informally, that an expression that is part of some complex expression can be substituted with an expression with the same semantic value without altering the semantic value of the complex expression. Note that substitution is an operation carried out at meta level and not in the logic itself. In this section we cover some concepts relating to substitution for L_1. As always, these concepts can be extended to any other formal language L.

Definition 3.14. Let $\varphi \in \mathsf{Formulae}$ and $A \in \mathsf{Terms} \cup \mathsf{PredSymb} \cup \mathsf{Formulae}$. We define the *free occurrences* of A in φ by structural induction on φ, w.r.t. Definition 3.1.

1. If φ is an atomic formula, i.e., $\varphi = P(t_1, \ldots, t_n)$, where $P \in \mathsf{PredSymb}_n$ and $t_1, \ldots, t_n \in \mathsf{Terms}$, then any occurrence of A in φ is free, that is:
   (a) If $A = \varphi$, then this occurrence of A is free
   (b) If for some $j = 1, \ldots, n$, we have $t_j = A$, then this occurrence of A is free
   (c) If $A = P$, then this occurrence of A is free
2. If $\varphi = \neg\psi$, then an occurrence of A in φ is free iff $A = \varphi$ or this is a free occurrence of A in ψ
3. If $\varphi = (\psi * \xi)$ and $*$ is either $\wedge$, $\vee$, $\to$ or $\leftrightarrow$, then an occurrence of A in φ is free iff $A = (\psi * \xi)$, or this is a free occurrence of A either in ψ or in ξ
4. If $\varphi = Qx\,\psi$, where Q is either $\exists$ or $\forall$ and $x \in \mathsf{Vars}$, then an occurrence of A in φ is free iff this is a free occurrence of A in ψ and $x \notin \mathsf{FreeV}(A)$, or $A = Qx\,\psi$
5. (This is a special case of atomic formulae, as in item 1.) If $\varphi = (t_1 = t_2)$, where $t_1, t_2 \in \mathsf{Terms}$, then an occurrence of A in φ is free iff $A = t_1$, $A = t_2$ or $A = \varphi$.

Informally, what Definition 3.14 says is that if an expression A occurs in a formula φ, then it is free as long as there is no variable occurring in A that is bound by some quantifier occurring in φ but not in A. I stress again that this definition applies to L_1 but can of course be modified for any language L.

Definition 3.15. If $\varphi \in \mathsf{Formulae}$ and $A \in \mathsf{Terms} \cup \mathsf{PredSymb} \cup \mathsf{Formulae}$, we say that an occurrence of A in φ is *bound* iff it is not free.

Notation 7. The result of replacing all *free occurrences of* A in φ with B is denoted by (35):

$$\varphi\{\, A :\equiv B \,\} \tag{35}$$

Notation 8. Let $\varphi \in \mathsf{Formulae}$ and, for $i = 1, \ldots, n$ $(n \geq 1)$, let A_i, B_i be well-formed expressions of L, i.e., for $L_1, A_i, B_i \in \mathsf{Const} \cup \mathsf{FreeV} \cup \mathsf{PredSymb} \cup \mathsf{Formulae}$, with A_i pairwise distinct. The result of the simultaneous replacement of all *free occurrences of all* A_i in φ, correspondingly with B_i, is denoted by (36):

$$\varphi\{\, A_1 :\equiv B_1,\ \ldots,\ A_n :\equiv B_n \,\} \tag{36}$$

Definition 3.16 (Free Substitution)**.** A *substitution* (36) *is free* iff (37) holds for all $i = 1, \ldots, n$ $(n \geq 1)$:

$$y \in \mathsf{FreeV}(B_i) \implies y \in \mathsf{FreeV}\,(\varphi\{\, A_1 :\equiv B_1,\ \ldots,\ A_n :\equiv B_n \,\}) \tag{37}$$

Theorem 3.2 (Equivalent Substitutions)**.** Assume that $\varphi \in \mathsf{Formulae}$ and, for $i = 1, \ldots, n$ $(n \geq 1)$, $A_i, B_i \in \mathsf{Const} \cup \mathsf{FreeV} \cup \mathsf{PredSymb} \cup \mathsf{Formulae}$, with A_i pairwise distinct. Let

$$\varphi^* = \varphi\{\, A_1 :\equiv B_1,\ \ldots,\ A_n :\equiv B_n \,\} \tag{38}$$

be a free substitution where:

$$[\![A_i]\!]^{\mathcal{M},h} = [\![B_i]\!]^{\mathcal{M},h}, \quad \text{for all } i = 1, \ldots, n \tag{39}$$

Then, we have $[\![\varphi]\!]^{\mathcal{M},h} = [\![\varphi^*]\!]^{\mathcal{M},h}$.

Proof. We show the statement by induction on φ:

Base Case

1. $\varphi = P(t_1, \ldots, t_n)$ where $P \in \mathsf{PredSymb}_n$ and $t_1, \ldots, t_n \in \mathsf{Terms}$. Consider $\varphi^* = \varphi\{\, P :\equiv Q,\ t_1 :\equiv s_1,\ \ldots,\ t_n :\equiv s_n \,\}$ where $Q \in \mathsf{PredSymb}_n$ and $s_1, \ldots, s_n \in \mathsf{Terms}$. Note that some of these replacements might be of the exact same expression (and so we cover all cases of substitution). Assume further that:

$$[\![P]\!]^{\mathcal{M},h} = [\![Q]\!]^{\mathcal{M},h} \tag{40a}$$

$$[\![t_i]\!]^{\mathcal{M},h} = [\![s_i]\!]^{\mathcal{M},h} \qquad (i = 1, \ldots, n) \tag{40b}$$

We want to show that $[\![\varphi]\!]^{\mathcal{M},h} = [\![\varphi^*]\!]^{\mathcal{M},h}$. For convenience we define the function $h^* : \mathsf{Terms} \to D$ in (41):

$$h^*(\alpha) = \begin{cases} h(\alpha) & \text{if } \alpha \in \mathsf{Vars} \\ I(\alpha) & \text{if } \alpha \in \mathsf{Const} \end{cases} \tag{41}$$

This results in the following reformulation of the truth conditions for $P(t_1, \ldots, t_n)$ and $Q(s_1, \ldots, s_n)$:

$$[\![P(t_1, \ldots, t_n)]\!]^{\mathcal{M},h} = 1 \iff (h^*(t_1), \ldots, h^*(t_n)) \in [\![P]\!]^{\mathcal{M},h} \tag{42}$$

$$[\![Q(s_1, \ldots, s_n)]\!]^{\mathcal{M},h} = 1 \iff (h^*(s_1), \ldots, h^*(s_n)) \in [\![Q]\!]^{\mathcal{M},h} \tag{43}$$

Also, our construction of h^* gives us, by (40b):

$$h^*(t_i) = h^*(s_i) \tag{44}$$

For $i = 1, \ldots, n$. The statement now follows immediately from (40a) and (42)–(44):

$$[\![P(t_1, \ldots, t_n)]\!]^{\mathcal{M},h} = 1 \iff (h^*(t_1), \ldots, h^*(t_n)) \in [\![P]\!]^{\mathcal{M},h} \tag{45a}$$

$$\iff (h^*(s_1), \ldots, h^*(s_n)) \in [\![Q]\!]^{\mathcal{M},h} \tag{45b}$$

$$\iff [\![Q(s_1, \ldots, s_n)]\!]^{\mathcal{M},h} = 1 \tag{45c}$$

The above is what we wanted to show for the base case.

Induction Step Assume as induction hypothesis that the statement holds for all ψ less complex (with respect to the inductive definition of L_1-formulae) than φ.

4. $\varphi = \neg\psi$

(Case 1) $A_1 = \varphi$ ($n = 1$), i.e., the only occurrence of A_1 is φ. Then, the only free substitution is:

$$\varphi^* = \varphi\{\, A_1 :\equiv B_1 \,\} = B_1, \tag{46}$$

where, by (39), $[\![A_1]\!]^{\mathcal{M},h} = [\![B_1]\!]^{\mathcal{M},h}$. Thus, we have:

$$[\![\varphi]\!]^{\mathcal{M},h} = [\![A_1]\!]^{\mathcal{M},h} = [\![B_1]\!]^{\mathcal{M},h} = [\![\varphi^*]\!]^{\mathcal{M},h} \tag{47}$$

Therefore, the statement holds.

(Case 2) All the free occurrences of A_i ($i = 1, \ldots, n$) are in ψ. The equivalence (48b) holds by the induction hypothesis. Thus, we have:

$$[\![\varphi]\!]^{\mathcal{M},h} = 1 \iff [\![\psi]\!]^{\mathcal{M},h} = 0 \tag{48a}$$

$$\underset{\text{ind.}}{\iff} [\![\psi\{\, A_1 :\equiv B_1,\ \ldots,\ A_n :\equiv B_n \,\}]\!]^{\mathcal{M},h} = 0 \tag{48b}$$

$$\iff [\![\neg\psi\{\, A_1 :\equiv B_1,\ \ldots,\ A_n :\equiv B_n \,\}]\!]^{\mathcal{M},h} = 1 \tag{48c}$$

$$\iff [\![\varphi^*]\!]^{\mathcal{M},h} = 1 \tag{48d}$$

Therefore, $[\![\varphi]\!]^{\mathcal{M},h} = [\![\varphi^*]\!]^{\mathcal{M},h}$, and the statement holds in this case too.

5. $\varphi = (\psi \wedge \xi)$. The case when $A = \varphi$ is proved as in (Case 1) above. We show the second case:

$$\begin{aligned}
[\![\varphi]\!]^{\mathcal{M},h} = 1 &\iff [\![\psi]\!]^{\mathcal{M},h} = [\![\xi]\!]^{\mathcal{M},h} = 1 && \text{(49a)}\\
&\underset{\text{ind.}}{\iff} [\![\psi\{\, A_1 :\equiv B_1,\ \ldots,\ A_n :\equiv B_n \,\}]\!]^{\mathcal{M},h} = \\
&\qquad [\![\xi\{\, A_1 :\equiv B_1,\ \ldots,\ A_n :\equiv B_n \,\}]\!]^{\mathcal{M},h} = 1 && \text{(49b)}\\
&\iff [\![\psi\{\, A_1 :\equiv B_1,\ \ldots,\ A_n :\equiv B_n \,\}\wedge \\
&\qquad \xi\{\, A_1 :\equiv B_1,\ \ldots,\ A_n :\equiv B_n \,\}]\!]^{\mathcal{M},h} = 1 && \text{(49c)}\\
&\iff [\![(\psi \wedge \xi)\{\, A_1 :\equiv B_1,\ \ldots,\ A_n :\equiv B_n \,\}]\!]^{\mathcal{M},h} = 1 && \text{(49d)}\\
&\iff [\![\varphi^*]\!]^{\mathcal{M},h} = 1 && \text{(49e)}
\end{aligned}$$

6. The statement for the cases when $\varphi = (\psi \vee \xi)$, $\varphi = (\psi \to \xi)$, $\varphi = (\psi \leftrightarrow \xi)$, and $\varphi = (\psi = \xi)$ is proved in a similar manner as in 5

7. $\varphi = \forall x\, \psi$. The case when $A = \varphi$ is again proved as in (Case 1) above. We show the second case:

$$\begin{gathered}
[\![\varphi]\!]^{\mathcal{M},h} = 1 \iff [\![\psi]\!]^{\mathcal{M},h'} = 1 \\
\text{for all assignments } h' \text{ s.t. if } u \in \mathsf{Vars} - \{x\} \text{ then } h(u) = h'(u)
\end{gathered} \tag{50}$$

   We also have:

$$\begin{gathered}
[\![\varphi^*]\!]^{\mathcal{M},h} = 1 \iff [\![\psi\{\, A_1 :\equiv B_1, \ldots, A_n :\equiv B_n \,\}]\!]^{\mathcal{M},h'} = 1 \\
\text{for all assignments } h' \text{ s.t. if } u \in \mathsf{Vars} - \{x\} \text{ then } h(u) = h'(u)
\end{gathered} \tag{51}$$

   Since the substitution is free, we know that no new occurrences of x have been added by the substitution. Hence, for any h' in (50)–(51) we have by the induction hypothesis that:

$$[\![\psi]\!]^{\mathcal{M},h'} = 1 \iff [\![\psi\{\, A_1 :\equiv B_1, \ldots, A_n :\equiv B_n \,\}]\!]^{\mathcal{M},h'} = 1 \tag{52}$$

   which proves the statement

8. The statement for the case when $\varphi = \exists x\, \psi$ is proved in the same manner as in 7

This shows that:

$$[\![\varphi]\!]^{\mathcal{M},h} = [\![\varphi\{\, A_1 :\equiv B_1, \ldots, A_n :\equiv B_n \,\}]\!]^{\mathcal{M},h} \tag{53}$$

which is what we wanted to show. □

Theorem 3.2 is crucial if we want our system to respect compositionality. That is, if we give two expressions the same semantic value, they should be able to play the exact same role in a sentence. We note that the theorem does not hold if we omit the requirement that the substitution in question is free, as shown in Example 3.2.

Example 3.2. Let $P \in \mathsf{PredSymb}_2$, $x \in \mathsf{Vars}$, $c \in \mathsf{Const}$ and $\mathcal{M} = (D, I)$ be such that $D = \{a, b\}$, $I(P) = \{(a, b)\}$ and $I(c) = b$. The following then holds:

$$[\![\exists x\, P(x, c)]\!]^{\mathcal{M},h} = 1 \tag{54}$$

for all assignments h. Let h' be an assignment such that:

$$[\![x]\!]^{\mathcal{M},h'} = h'(x) = b = [\![c]\!]^{\mathcal{M},h'} \tag{55}$$

Now consider:

$$\psi = [\exists x\, P(x, c)]\{\, c :\equiv x \,\} = \exists x\, P(x, x) \tag{56}$$

for which we have:

$$[\![\psi]\!]^{\mathcal{M},h'} = [\![\exists x\, P(x, x)]\!]^{\mathcal{M},h'} = 0 \neq 1 = [\![\exists x\, P(x, c)]\!]^{\mathcal{M},h'} \tag{57}$$

which, of course, would contradict any theorem about a general substitution of bound variables.

3.4 Advantages of L_1

Our language L_1 can be used to formalise large parts of HL. In Example 3.1, $Pa \wedge Qa$ can be though of as capturing "Edith is a woman and Edith smokes" or, perhaps, "Edith is a smoking woman". In the same way $Qa \to \exists x\, R(x, c)$ captures the somewhat odd sentence "If Edith is a woman, then someone is taller than Jacques". The idea is that the L_1-versions of HL sentences capture their logical form — their underlying grammatical structure.

There are some strong advantages to analysing HL using L_1. First, in L_1, every sentence has a truth value. Consider the following sentence:

S1 The king of France is bald [3]

Intuitively, either the sentence is true or it is false. But there is no king of France, so how do we determine which answer is correct? This problem was one of the earliest motivators for the use of L_1 in the treatment of English language sentences. Russell [15] realised that a formalisation in L_1 of the sentence made its truth value clear. We let:

$$\text{king of France} \xrightarrow{\mathsf{render}} \mathbf{King\text{-}F} \tag{58}$$

and

$$\text{is bald} \xrightarrow{\mathsf{render}} \mathbf{bald} \tag{59}$$

The sentence can now be rendered as:

$$\exists x(\mathbf{King\text{-}F}(x) \wedge \forall y(\mathbf{King\text{-}F}(y) \to y = x) \wedge \mathbf{bald}(x)) \tag{60}$$

[3] For a more mathematical example, consider the statement "The largest prime number is not a perfect square". Is it true or false?

The second perhaps slightly confusing conjunct is there to capture the uniqueness entailed by the determiner "the" in "the king of France". Now, it is not difficult to calculate the truth value of the sentence. Relative to a model $\mathcal{M} = (D, I)$ where there is no king of France (that is, $I(\textbf{King-F}) = \varnothing$) the first conjunct will always be false, and consequently the whole sentence is false.

Secondly, L_1 can capture ambiguities in HL that do not seem to let themselves be captured otherwise. The most common problem is that of quantifier ambiguities. Let us consider a simple example from mathematics:

S2 Every integer is greater than some rational number

It is by no means clear what is claimed in S2. There are two possible answers:

A1 The proposition is true. Let $n \in \mathbb{Z}$. Define $i_n = n - 1$. Clearly, $i_n \in \mathbb{Q}$ and $i_n < n$

A2 The proposition is false. Assume that every integer is greater than some $q \in \mathbb{Q}$. Then $\lfloor q \rfloor \in \mathbb{Z}$ and $\lfloor q \rfloor < q$

The reason that these two contradictory arguments both seem to work is that the sentence can be seen as the representation of two different L_1-sentences:

A1* $\forall x\,(\textbf{integer}(x) \to \exists y\,(\textbf{rational}(y) \wedge \textbf{greater-than}(x, y)))$

A2* $\exists x\,(\textbf{rational}(x) \wedge \forall y\,(\textbf{integer}(y) \to \textbf{greater-than}(y, x)))$

These two forms explain how the two answers can look to different. Should we answer as in A1, we do so because we have analysed the sentence as in A1*. An alternative way of phrasing it is to say that "every" (i.e. "$\forall$") has been given a *wide* scope, whereas "some" (i.e. "$\exists$") has been given a *narrow* scope. Should we, conversely, answer as in A2, we do so because we have analysed the sentence as in A2*. Here, "every" has been given a narrow scope, whereas "some" has been given a wide scope. We refer to the A1/A1*-reading as the *de dicto*-reading and to the A2/A2*-reading as the *de re*-reading. Unsurprisingly, de dicto-readings are far more common than de re-readings.

3.5 Complex Individual Terms

One way to improve the renderings of human language into FOL is to allow for function symbols. When we allow for functions from individual terms to individual terms we are provided with a much more systematic way to treat otherwise complex expressions involving quantifiers. The functions also provide us with a way of directly referring to the objects that we want to talk about, rather than describing them by a complex clause of quantifiers. For example, we could get the following rendering of (60) (ignoring any philosophical questions this particular sentence may raise):

$$\text{The King of France is bald} \xrightarrow{\textsf{render}} \textbf{bald}(f_{\textsf{THE-KING-OF}}(\textbf{France})) \qquad (61)$$

The rendering in (61) is clearly simpler than (60). Here $f_{\textsf{THE-KING-OF}}$ can be thought of as representing a function that takes the object denoted by **France** as input to then output one unique individual (i.e., the King of France). Note that rather than saying something like "there exists an object such that it is such and such", like we did in (60), we directly refer to the individual denoted by $f_{\textsf{THE-KING-OF}}(\textbf{France})$.

It is however not entirely clear whether this is a satisfactory rendering, as $f_{\textsf{THE-KING-OF}}$ appears to represent some object with the structure NP P, which is not permitted by our PS rules (9a)–(9u). We want our rendering to reflect that "of" is combined with "France" to create the PP "of France" which can *then* be combined with the NP "The King". This is not what happens in (61).

To get a FOL with functions, L_1^{fun}, we simply extend L_1 by adding a set of function symbols $\textsf{FunSymb}_{L_1^{\text{fun}}} = \{f_0^{i_0}, f_1^{i_1}, \ldots, f_m^{i_m}\}$ for a fixed $m \in \mathbb{N}$ where for each $j = 0, \ldots, m$, $f_j^{i_j}$ is a function symbol of arity $i_j \in \mathbb{N}$.

Let $c \in \textsf{Const}_{L_1}$, $x \in \textsf{Vars}$, $f \in \textsf{FunSymb}_{L_1^{\text{fun}}}$, and $T, t_1, \ldots, t_n \in \textsf{Terms}_{L_1^{\text{fun}}}$. Then the set of $\textsf{Terms}_{L_1^{\text{fun}}}$ is defined by induction, in (62), in BNF-style:

$$\begin{aligned} T &:\equiv c \mid x \mid f^n(t_1, \ldots, t_n) \\ &\quad \text{for } c \in \textsf{Const},\ x \in \textsf{Vars},\ f^n \in \textsf{FunSymb}_{L_1^{\text{fun}}}, \\ &\quad t_i \in \textsf{Terms}_{L_1^{\text{fun}}},\ i \in \{1, \ldots, n\},\ n \geq 0 \end{aligned} \tag{62}$$

The set $\textsf{Formulae}_{L_1^{\text{fun}}}$ is defined in the same way as it was for L_1, by adding (62), and it clearly extends $\textsf{Formulae}_{L_1}$ since $\textsf{Terms}_{L_1} \subset \textsf{Terms}_{L_1^{\text{fun}}}$.

3.6 Limitations of First-Order Logics

In the following sections I shall mention a few of the problems that arise when we try to render into FOL. I will focus on examples from L_1. This is without any loss of generality, since the problems extend to L_1^{fun} and all other FOLs.

One limitation that I will not mention here is coordination of VPs, NPs and Adjs. This will be covered in depth in Section 4.5.2.

3.6.1 Predicate Modification

It is not difficult to see that many of our phrase structure rules will not work when implemented in L_1. Let us show this. Consider the following two sentences:

S3 Jacques sings

S4 Jacques sings well

Assume we are to formulate these sentences in L_1. Here is one approach for S3:

1. The whole sentence is on the form NP VP and must be represented by a sentence φ in L_1

2. The verb "sings" takes "Jacques" as an argument. The only objects in L_1 that can take an argument and output a sentence are those in $\mathsf{PredSymb}$. Since it only takes one argument, it must be of arity 1. So:

$$\text{sings} \xrightarrow{\mathsf{render}} \mathbf{sing} \in \mathsf{PredSymb}_1 \tag{63}$$

3. The name "Jacques" is an argument of **sing**. The only objects in L_1 that can combine with a predicate and result in a sentence are those in Const. So:

$$\text{Jacques} \xrightarrow{\mathsf{render}} \mathbf{j} \in \mathsf{Const} \tag{64}$$

By the above, it is clear that the correct rendering of S3 is:

$$\text{Jacques sings} \xrightarrow{\mathsf{render}} \mathbf{sing}(\mathbf{j}) \tag{65}$$

What then is the correct rendering of S4? The sentence is on the form NP VP. So, as we want uniformity with the rendering of S3, we get the rendering in (64) as well as:

$$\text{sings well} \xrightarrow{\mathsf{render}} \mathbf{sing\text{-}well} \in \mathsf{PredSymb}_1 \tag{66}$$

So the rendering of S4 should be:

$$\mathbf{sing\text{-}well}(\mathbf{j}) \tag{67}$$

But this rendering contradicts PS Rule (9l). We have $\mathbf{sing\text{-}well} \in \mathsf{PredSymb}_1$ and consequently it cannot be analysed further. In other words, there is no way of separating **sing** from **well**.

There is one other possible approach. One could try to to render S4 as:

$$\mathbf{sing}(j) * \mathbf{well}(j) \tag{68}$$

where $*$ is $\wedge, \vee, \rightarrow$ or $\leftrightarrow$. But this too is absurd, since $\mathbf{well}(\mathbf{j})$ appears to be on the form NP Adv, which is not permitted by our grammar. So S4 does not appear to have a satisfactory rendering in L_1.

The problem can be summed up as follows. In L_1, there is no way for predicate symbols to take other predicate symbols as arguments, which is something HL seems to require. Just like a verb appears to take a noun phrase as an argument,[4] we need to let adverbs be able to take verbs as arguments if we are to make sense of them. This is one of the primary motivation for the need of a higher order language.

[4]Although we shall see later that this is not necessarily the case.

3.6.2 Quantification

Previously, we saw that one of the strengths of L_1 was that it could make sense of the ambiguities in sentences with multiple quantifiers. Unfortunately, it is also its treatment of quantification that is one of its major drawbacks. Consider the following sentence:

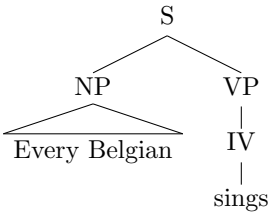
S5 Every Belgian sings

If we want to render S5 into L_1, should get the following rendering:

$$\text{Every Belgian sings} \xrightarrow{\text{render}} \forall x(\mathbf{belgian}(x) \rightarrow \mathbf{sing}(x)) \tag{69}$$

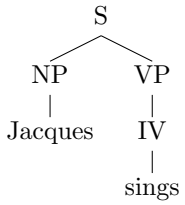
Let us consider the corresponding phrase structure of this sentence:

(70)



Compare this to the phrase structure of S3:

(71)



Clearly, the structures of these two trees are reminiscent of each other. On this level, they are in fact the same. Herein lies the problem — we need to explain how (69) can describe the correct rendering of S5 when S3 renders $\mathbf{sing}(\mathbf{j})$. "Every Belgian" appears to have on the form $\forall x(\mathbf{belgian}(x) \rightarrow *)$ where $*$ somehow needs to be replaced by $\mathbf{sing}(x)$. There are two problems with this. First, when rendering $\mathbf{sing}(\mathbf{j})$ from S3, we did so by applying the VP on the NP. Here, if $*$ is to be replaced by $\mathbf{sing}(x)$, we need to apply the NP on the VP which of course contradicts the uniformity of our rendering. The second problem is that there simply is no mechanism in L_1 that allows us to replace $*$ with $\mathbf{sing}(x)$. As we shall see later we can do this using the tools of simply typed lambda calculus, but we are not there yet.

We should also note that it gets even more problematic when it comes to dealing with sentences like S2, which have renderings that contain multiple quantifiers. Here, the rendering of the second NP will contain a variable from the first NP, despite their renderings being independent of each other. In L_1, it is difficult to make sense of this.

3.6.3 Tense

One perhaps less fundamental limitation of L_1 is that it is bad at expressing statements that are relative to time. Let us consider two similar sentences to S3:

S6 Jacques sang

S7 Jacques will sing

Let us try to render S6 and S7 into L_1. The first approach is to simply introduce new predicates $\mathbf{Sing\text{-}P}, \mathbf{Sing\text{-}F} \in \mathsf{PredSymb}_1$ so that:

$$\text{Jacques sang} \xrightarrow{\mathsf{render}} \mathbf{sing\text{-}P}(\mathbf{j}) \tag{72}$$

and

$$\text{Jacques will sing} \xrightarrow{\mathsf{render}} \mathbf{sing\text{-}F}(\mathbf{j}) \tag{73}$$

This might work in some cases. However, there are several problems with this approach. First, it treats the different tenses of the verb as having nothing in common. The above rendering does not capture that "sang" and "will sing" in fact correspond to the same verb, but tensed differently. This is perhaps, mathematically speaking, a minor point but for the linguist it should be worrisome. We do not just want our rendering to work from a technical point of view — we want it to capture in a clear way how HL works. I claim that this is not done by the above renderings.

It is, in fact, possible to translate any tense logic into FOL, see Garson [4]. The technicalities of such a translation is beyond the scope of this essay (as the translations shall not be used), but it is worth noting that these translations still require us first to render our English language sentences into some system of temporal logic before being able to translate them into L_1.

3.6.4 Modality

We wish to give adequate renderings of words such as "necessarily" and "possibly". This discussion is essentially the same as that concerning tense.

As we saw in section 2.1, the semantic values of these expressions are though of in terms of *possible worlds*. Intuitively, a possible world is copy of our current world where things have been altered in a non-contradictory way. Mathematically, two possible world w_1, w_2 are simply indices such that if a is some expression in L_1, then it is possible that $[\![a]\!]^{\mathcal{M},w_1} \neq [\![a]\!]^{\mathcal{M},w_1}$. So by saying that something is necessarily true we mean that it is true in all possible worlds. By saying that something is possibly true we mean, on the other hand, that it is true in at least one possible world. We notice here the similarity between these two expressions and the tensed verbs. Saying "Jacques sang" is true is to say that there exists one point in time (in the past) where "Jacques sings" is true. Mathematically speaking, points in time and possible worlds are the same type of objects, and so the reason why L_1 is limited in its treatment of "necessarily" and "possibly" is the same as why it is limited in its treatment of tensed verbs.

3.6.5 Intensionality

We start of this section by the following definition:

Definition 3.17. We say a context introduced by a sentence S is *extensional* if for any word a occurring in the sentence, a can be replaced with a co-referring word b (assuming the replacement respects the syntax of the language in question) without changing the semantic value of S. If the context is not extensional, we call it *intensional*.

For an example of intensionality, consider the sentence:

S8 Martin Heidegger wrote silly books

Since the sentence is true, and since "Martin Heidegger" refers to the same thing as "The author of Being and Time"[5], we can make a substitution and end up with:

S9 The author of Being of Time wrote silly books

Both S8 and S9 are true sentences, so the context appears to be extensional.

Extensionality does not, however, always hold. There are some words that introduce a context where extensionality no longer holds. Suppose there is a young aspiring logician, Ludwig, who is not as well-versed in the fascinating school of continental philosophy as we are. Ludwig only knows that Martin Heidegger wrote some silly books, but does not know Being and Time is one of them. Then:

S10 Ludwig thinks that Martin Heidegger wrote silly books

is a true sentence, whereas:

S11 Ludwig thinks that the author of Being and Time wrote silly books

is false. Let us now consider the renderings in L_1 of S10 and S11. Let:

$$\text{silly book} \xrightarrow{\mathsf{render}} \textbf{silly-book} \tag{74a}$$

$$\text{... thinks...wrote...} \xrightarrow{\mathsf{render}} \textbf{thinks-wrote} \tag{74b}$$

$$\text{Ludwig} \xrightarrow{\mathsf{render}} \mathbf{l} \tag{74c}$$

$$\text{Martin Heiddeger} \xrightarrow{\mathsf{render}} \mathbf{m} \tag{74d}$$

$$\text{The author of Being and Time} \xrightarrow{\mathsf{render}} \mathbf{m}' \tag{74e}$$

Clearly, in our intended model $\mathcal{M}$, $[\![\mathbf{m}]\!]^{\mathcal{M}} = [\![\mathbf{m}']\!]^{\mathcal{M}}$. We should get something along the lines of:

Ludwig thinks that Martin Heidegger wrote silly books (75a)

[5]Strictly speaking, it could be argued that a citation "A" refers to the word A. We ignore these language-philosophical matters here (and prioritise readability). For details, see Russell [15]

$$\xrightarrow{\text{render}} \exists x(\textbf{silly-book}(x) \wedge \textbf{thinks-wrote}(\mathbf{l}, \mathbf{m}, x)) \tag{75b}$$

Ludwig thinks that the author of Being and Time wrote silly books (75c)

$$\xrightarrow{\text{render}} \exists x(\textbf{silly-book}(x) \wedge \textbf{thinks-wrote}(\mathbf{l}, \mathbf{m}', x)) \tag{75d}$$

But since $[\![\mathbf{m}]\!]^{\mathcal{M}} = [\![\mathbf{m}']\!]^{\mathcal{M}}$, Theorem 3.2 gives us (76):

$$\begin{aligned} &[\![\exists x(\textbf{silly-book}(x) \wedge \textbf{thinks-wrote}(\mathbf{l}, \mathbf{m}, x))]\!]^{\mathcal{M}} = 1 \\ \iff &[\![\exists x(\textbf{silly-book}(x) \wedge \textbf{thinks-wrote}(\mathbf{l}, \mathbf{m}', x))]\!]^{\mathcal{M}} = 1 \end{aligned} \tag{76}$$

So both rendering are given identical truth-conditions, despite S10 being true and S11 being false. Since the truth conditions of the two sentences are different, we shall require a rendering that somehow reflects that "thinks that" introduces an intensional context.

4 Montague Intensional Logic

We now want to extend L_1 to a language that can better handle the previously mentioned limitations. The new language will have 4 properties:

- P1 It will be a *type theoretic* language
- P2 Every predicate will be represented by a unary function
- P3 It will be a *higher-order language*. That is, it will allow for abstraction over predicates, rather than just individuals
- P4 It will contain mechanisms for making sense of tense, modality and intensionality

At first, in Example 4.1, we revisit L_1 and reformulate a part of it to satisfy P1 and P2. Then, we extend this idea to the syntax of L_{IL}.

Example 4.1. Our goal is to extend L_1 by adding *types* and restricting our non-logical constants to these types. I.e., we assign a type for indivduals to the members of Const_{L_1} and types for unary functions to the members of $\mathsf{PredSymb}_{L_1}$. Let us limit ourselves to unary and binary predicates, which we will represent by appropriate unary functions.

(i) If $\alpha \in \mathsf{Const}$, then α is of type e

(ii) If $\varphi \in \mathsf{Formulae}$, then φ is of type t

(iii) If $P \in \mathsf{PredSymb}$ and P is of arity 1, then P is of type $\langle e, t\rangle$

(iv) If $R \in \mathsf{PredSymb}$ and R is of arity 2, then R is of type $\langle e, \langle e, t\rangle\rangle$

We can also do the same for the logical constants[6]:

[6] We do not do this for $=$, since this requires more sophisticated methods

(v) If $\alpha = \neg$, then α is of type $\langle t, t\rangle$

(vi) If $\alpha \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, then α is of type $\langle t, \langle t, t\rangle\rangle$

This is, of course, not yet precise. Before we give a more rigorous and general definition of types, it is good to get a basic intutition of what (i)–(vi) mean. The types essentially work as syntactic categories for expressions of our language. They also determine what kind of objects the expressions denote. When we say that α and φ are of types e and t respectively, we are (1) specifying what kind of objects they denote (i.e. α denotes an individual, φ denotes a truth value) and (2) representing some of their syntactic properties. What we mean by (2) is that the types restrict what strings of expressions we are allowed to form. For example, if P is a well-formed expression of type $\langle e, t\rangle$, then the idea is that P can be combined with another well-formed expression of type e and that this will then produce a more complex expression of type t (we shall see how this works in practice later in this section). It is an important feature that P can *only* be combined with well-formed expressions of type e. I.e., if α is of type e and φ is of type t, then $P(\alpha)$ is a well-formed expression of type t, whereas $P(\varphi)$ is not well-formed.

4.1 Syntax

I hope that Example 4.1 has given a fairly clear idea of what types are and how they can be used. We are now ready to do things formally. First, we introduce the following notational agreements:

Notation 9. If a is of type A we write $a : A$. If several expressions are of the same type, e.g., $a : \tau$ and $b : \tau$, we write $a, b : \tau$.

Notation 10. Instead of writing $*(b)(a)$, where $*$ is a logical constant of type $\langle t, \langle t, t\rangle\rangle$ and $a, b : t$, we use the infix notation: $(a * b)$. This allows us to write conjuction and disjunction as $(a \wedge b)$ and $(a \vee b)$ rather than $\wedge(b)(a)$ and $\vee(b)(a)$.

The syntax of L_{IL} is given by Definition 4.1. Originally, L_{IL} was introduced by Montague [12] by using it for a formal grammar for translating a fragment of English Language in it.

Definition 4.1 (Syntax of L_{IL})**.**

1. The set Types of the types of L_{IL} is defined by the recursive rules (77) in BNF:

$$\tau ::= e \mid t \mid \langle s, \tau\rangle \mid \langle \tau, \tau\rangle \tag{77}$$

   Note that the symbol s is not a type, but a special syntactic object, which is used in composition of complex types. It is used to signify possible worlds

2. The following are the basic expressions of L_{IL}:

(i) If $\sigma \in \mathsf{Types}$, then $\mathsf{Const}_\sigma :\equiv \{c_0^\sigma, c_1^\sigma, \dots\}$ is a countable set of non-logical constants of type σ

(ii) If $\sigma \in \mathsf{Types}$, then $\mathsf{Vars}_\sigma :\equiv \{x_0^\sigma, x_1^\sigma, \dots\}$ is a countable set of variables of type σ

(iii) *Logical Constants:* $\neg : \langle t, t\rangle$ and $\wedge, \vee, \to, \leftrightarrow : \langle t, \langle t, t\rangle\rangle$. We define $\mathsf{LogConst}_{\langle t,t\rangle} :\equiv \{\neg\}$ and $\mathsf{LogConst}_{\langle t,\langle t,t\rangle\rangle} :\equiv \{\wedge, \vee, \to, \leftrightarrow\}$

3. For each $\delta \in \mathsf{Types}$, the set ME_δ is the smallest set satisfying the following recursive rules:

(i) For every $\sigma \in \mathsf{Types}$, if $a \in \mathsf{Const}_\sigma \cup \mathsf{Vars}_\sigma$, then $a \in ME_\sigma$

(ii) $\neg \in ME_{\langle t,t\rangle}$ and $\wedge, \vee, \to, \leftrightarrow \in ME_{\langle t,\langle t,t\rangle\rangle}$

(iii) If $a \in ME_\tau$ and $x \in \mathsf{Vars}_\sigma$, then $(\lambda x\, a) \in ME_{\langle \sigma,\tau\rangle}$

(iv) If $a \in ME_{\langle \sigma,\tau\rangle}$ and $b \in ME_\sigma$, then $a(b) \in ME_\tau$

(v) If $a, b \in ME_\sigma$, then $(a = b) \in ME_t$

(vi) If $\varphi, \psi \in ME_t$ and $x \in \mathsf{Vars}_\sigma$ then $\forall x\, \varphi$, $\exists x\, \varphi$, $\mathbf{F}\,\varphi$, $\mathbf{P}\,\varphi$, $\Box\,\varphi \in ME_t$[7]

(vii) If $a \in ME_\sigma$, then $(\hat{}a) \in ME_{\langle s,\sigma\rangle}$[8]

(viii) If $a \in ME_{\langle s,\sigma\rangle}$, then $(\check{}a) \in ME_\sigma$ [9]

For each $\sigma \in \mathsf{Types}$, the set ME_σ is called the set of the *meaningful expressions* of type σ. The expressions $\varphi : t$ are called *formulae*. The expressions $(\lambda x\, a)$ are called λ-abstractions and $a(b)$ applications.

Definition 4.2.

$$\mathsf{Const} = \bigcup_{\tau \in \mathsf{Types}} \mathsf{Const}_\tau \tag{78a}$$

$$\mathsf{Vars} = \bigcup_{\tau \in \mathsf{Types}} \mathsf{Vars}_\tau \tag{78b}$$

$$\mathsf{LogConst} = \mathsf{LogConst}_{\langle t,t\rangle} \cup \mathsf{LogConst}_{\langle t,\langle t,t\rangle\rangle} \tag{78c}$$

$$ME = \bigcup_{\tau \in \mathsf{Types}} ME_\tau \tag{78d}$$

We make a few notational agreements and clarifications.

Notation 11. Parentheses will often be omitted when there is no risk for confusion. Note, in particular, the following cases:

1. Parentheses will often be omitted in terms with consecutive applications, by assuming *left association*. For instance, $(((P(x_1))(x_2))(x_3))(x_4)$ will be written as $P(x_1)(x_2)(x_3)(x_4)$

[7] For intuition, think of $\mathbf{F}\,\varphi$ as *in the future* φ, of $\mathbf{P}\,\varphi$ as *in the past* φ and of $\Box\,\varphi$ as *necessarily* φ

[8] For intuition, think of $(\hat{}a)$ as *intensionally* a

[9] For intuition, think of $(\check{}a)$ as *extensionally* a

2. If $(w,t) \in W \times T$ we omit one layer of parentheses when (w,t) is taken as an argument. E.g., we write $P(w,t)$ rather than $P((w,t))$

3. If we have multiple λ- abstractions following each other, we often use *right association* and omit the parenthesises. E.g., we sometimes write $\lambda P\, \lambda Q\, \varphi$ rather than $(\lambda P\, (\lambda Q\, \varphi))$

4. We often omit the parentheses in $(\check{}P)$

Notation 12. If $P : \sigma$ we sometimes write σ as a subscript of P, i.e. we write P_σ. This will only be done in cases where I feel it is important to note the type of an expression or where it increases readability.

Notation 13. Given $P : \langle s, \langle \sigma, \tau \rangle \rangle$ and $\alpha : \sigma$ the *brace notation for one argument* is given by (79):

$$P\{\alpha\} = (\check{}P)(\alpha) \tag{79}$$

For a version of this brace notation, see Montague [12]. Also, see Dowty et al. [3].

We also give the brace notation for two arguments:

Notation 14. Given $P : \langle s, \langle \sigma_1, \langle \sigma_2, \tau \rangle \rangle \rangle$, $\alpha_1 : \sigma_1$ and $\alpha_2 : \sigma_2$ the *brace notation for two arguments* is given by (80):

$$P\{\alpha_1, \alpha_2\} = (\check{}P)(\alpha_2)(\alpha_1) \tag{80}$$

Similarly to the case of L_1, the sets $\mathsf{FreeV}(\alpha)$ and $\mathsf{BoundV}(\alpha)$, for each $\alpha \in ME_\delta$ and $\delta \in \mathsf{Types}$, are defined by recursion, i.e., by structural induction on α, using Definition 4.1-3. We shall, however, not rely very heavily on these concepts in what follows. Consequently, a formal definition is omitted.

4.2 Semantics

The semantics of L_{IL} is given by Definition 4.3, again following Montague [12].

Notation 15. We shall use the following, typical notational agreement in (81):

$$B^A = \{\, f \mid f\colon A \to B \,\} \tag{81}$$

That is, B^A is the set of all total functions with domain A and codomain B.

The semantic values of the expressions of L_{IL} are assigned in models with a hierarchy of semantic sub-domains according to the types.

Definition 4.3. (Models of L_{IL}) A model of L_{IL} is a tuple $\mathcal{M} = (D, W, T, \leq, I)$, where: D is a nonempty set of objects, called *individuals* or *entities* (as in the models for L_1); W and T are nonempty sets, of *possible worlds* and *points in time*, respectively; $\leq$ is a linear ordering on T; I is the interpretation function

defined on the set Const of the constants of L_{IL}, which takes values that respect the type of the constants, according to (83a)–(83b). The ordered pairs (w, t), where $w \in W$ and $t \in T$, are called *indices*.

We introduce the concept of a *denotation* of an expression $A : \sigma$ to respect its type $\sigma \in \mathsf{Types}$. If $\sigma \in \mathsf{Types}$ then D_σ is the set of the possible denotations of the expressions A of type σ.

(i) $D_e = D$

(ii) $D_t = \{0, 1\}$

(iii) $D_{\langle\sigma,\tau\rangle} = D_\tau^{D_\sigma}$

(iv) $D_{\langle s,\sigma\rangle} = D_\sigma^{W\times T}$

The set $\mathbb{T}$ in (82) is called the *frame* of the model $\mathcal{M}$:

$$\mathbb{T} = \{D_\sigma \mid \sigma \in \mathsf{Types}\} \tag{82}$$

By the notational agreement (81), the sets $D_{\langle\sigma,\tau\rangle}$ and $D_{\langle s,\sigma\rangle}$, contain all of the corresponding functions. Because of this, the frame $\mathbb{T}$ and the models are called *standard*.

The interpretation function I of $\mathcal{M}$ is such that it satisfies (83a)–(83b):

$$I\colon \mathsf{Const} \to \{\, f \mid f\colon (W \times T) \to \cup\mathbb{T}\} \tag{83a}$$

$$I(c)(w, t) \in D_\sigma, \quad \text{for every } c \in \mathsf{Const}_\sigma,\ w \in W,\ t \in T \tag{83b}$$

Definition 4.4. (Extensions of the Meaningful Expressions of L_{IL}) Here we define the semantic values, which Montague called the *extensions* of the meaningful expressions, i.e., of the expressions that are elements of the set ME of L_{IL}. The semantic extension of every expression $\alpha \in ME_\tau$ is with respect to its type τ, and is relative to some $w \in W$, $t \in T$ and some assignment h. In general, the semantic values of the complex expressions may depend on the values of the constants in Const by the interpretation function I.

For every expression $\alpha \in ME$, we define its semantic value $[\![\alpha]\!]^{\mathcal{M},w,t,h}$, called the extension of α, by structural induction on α:

(i) If $\alpha \in \mathsf{Const}_\sigma$, for some $\sigma \in \mathsf{Types}$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = I(\alpha)(w, t)$

(ii) If $\alpha \in \mathsf{Vars}_\sigma$, for some $\sigma \in \mathsf{Types}$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = h(\alpha)$

(iii) If $\alpha = (\lambda x\, a)$, where $a \in ME_\sigma$ and $x \in \mathsf{Vars}_\tau$, for some $\sigma, \tau \in \mathsf{Types}$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![(\lambda x\, a)]\!]^{\mathcal{M},w,t,h} = g$, where g is a function on D_τ such that, for all $u \in D_\tau$, $g(u) = [\![a]\!]^{\mathcal{M},w,t,h'}$, where h' is just like h, possibly except on u, and $h'(x) = u$, i.e.:

$$h'(y) = \begin{cases} h(y), & \text{if } y \neq x \\ u, & \text{otherwise, i.e., if } y = x \end{cases} \tag{84}$$

(iv) If $\alpha = a(b)$ where $a \in ME_{\langle\sigma,\tau\rangle}$ and $b \in ME_\sigma$ for some $\sigma, \tau \in \mathsf{Types}$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![a(b)]\!]^{\mathcal{M},w,t,h} = [\![a]\!]^{\mathcal{M},w,t,h}([\![b]\!]^{\mathcal{M},w,t,h})$

(v) If $\alpha = \neg$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![\neg]\!]^{\mathcal{M},w,t,h}$ where, for any assignment g, $[\![\neg]\!]^{\mathcal{M},w,t,g} = \mathsf{neg} : \{0,1\} \to \{0,1\}$ is defined by $\mathsf{neg}(u) = 1 - u$

(vi) If $\alpha = \wedge$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![\wedge]\!]^{\mathcal{M},w,t,h}$ where, for any assignment g, $[\![\wedge]\!]^{\mathcal{M},w,t,g} = \mathsf{and} : \{0,1\} \to \{0,1\}^{\{0,1\}}$ is defined by $\mathsf{and}(u)(v) = \min\{u, v\}$, for all $u, v \in \{0,1\}$

(vii) If $\alpha = \vee$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![\vee]\!]^{\mathcal{M},w,t,h}$ where, for any assignment g, $[\![\vee]\!]^{\mathcal{M},w,t,g} = \mathsf{or} : \{0,1\} \to \{0,1\}^{\{0,1\}}$ is defined by $\mathsf{or}(u)(v) = \max\{u, v\}$ for all $u, v \in \{0,1\}$

(viii) If $\alpha = \to$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![\to]\!]^{\mathcal{M},w,t,h}$ where, for any assignment g, $[\![\to]\!]^{\mathcal{M},w,t,g} = \mathsf{imp} : \{0,1\} \to \{0,1\}^{\{0,1\}}$ is defined by $\mathsf{imp}(u)(v) = 1$ iff $v = 0$ or $u = 1$, for all $u, v \in \{0,1\}$[10]

(ix) If $\alpha = \leftrightarrow$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![\leftrightarrow]\!]^{\mathcal{M},w,t,h}$ where, for any assignment g, $[\![\leftrightarrow]\!]^{\mathcal{M},w,t,g} = \mathsf{iff} : \{0,1\} \to \{0,1\}^{\{0,1\}}$ is defined by $\mathsf{iff}(x)(y) = x + y + 1$ (in $\mathbb{Z}_2$)

(x) If α is $(a = b)$ where $a, b \in ME_\sigma$ for some $\sigma \in \mathsf{Types}$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![(a = b)]\!]^{\mathcal{M},w,t,h} = 1$ iff $[\![a]\!]^{\mathcal{M},w,t,h} \in D_\sigma$ is the same object as $[\![b]\!]^{\mathcal{M},w,t,h} \in D_\sigma$

(xi) If $\alpha = \forall x\, \varphi$ where $\varphi \in ME_t$ and $x \in \mathsf{Vars}_\sigma$ for some $\sigma \in \mathsf{Types}$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![\forall x\, \varphi]\!]^{\mathcal{M},w,t,h} = 1$ iff $[\![\varphi]\!]^{\mathcal{M},w,t,h'} = 1$, for all h' such that $h'(y) = h(y)$, for all $y \in \mathsf{Vars}_\sigma - \{x\}$

(xii) If $\alpha = \exists x\, \varphi$ where $\varphi \in ME_t$ and $x \in \mathsf{Vars}_\sigma$ for some $\sigma \in \mathsf{Types}$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![\forall x\, \varphi]\!]^{\mathcal{M},w,t,h} = 1$ iff $[\![\varphi]\!]^{\mathcal{M},w,t,h'} = 1$ for some h' where $h'(y) = h(y)$ for all $y \in \mathsf{Vars}_\sigma - \{x\}$

(xiii) If $\alpha = \Box\, \varphi$ where $\varphi \in ME_t$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![\Box\, \varphi]\!]^{\mathcal{M},w,t,h} = 1$ iff $[\![\varphi]\!]^{\mathcal{M},w',t',h} = 1$ for all $w' \in W$ and for all $t' \in T$

(xiv) If $\alpha = \mathbf{F}\, \varphi$ where $\varphi \in ME_t$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![\mathbf{F}\, \varphi]\!]^{\mathcal{M},w,t,h} = 1$ iff $[\![\varphi]\!]^{\mathcal{M},w,t',h} = 1$ for some $t' \in T$ such that $t' > t$

(xv) If $\alpha = \mathbf{P}\, \varphi$ where $\varphi \in ME_t$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![\mathbf{P}\varphi]\!]^{\mathcal{M},w,t,h} = 1$ iff $[\![\varphi]\!]^{\mathcal{M},w,t',h} = 1$ for some $t' \in T$ such that $t' < t$

(xvi) If $\alpha = (\,\hat{}\, a)$ where $a \in ME_\sigma$ for some $\sigma \in \mathsf{Types}$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![(\,\hat{}\, a)]\!]^{\mathcal{M},w,t,h}$ is a function g defined on $W \times T$, such that, for all $(w', t') \in W \times T$, $g(w', t') = [\![a]\!]^{\mathcal{M},w',t',h}$

[10] The order of the arguments in $\mathsf{imp}(u)(v) = 1$ might be unexpected. This is in fact the curried version of a binary function $\mathsf{imp}^*(v, u)$ defined just as $\mathsf{imp}(u)(v)$. Hence, this should be thought of as representing "v implies u", rather than u implies v, which the order of the arguments might entail.

(xvii) If $\alpha = (\check{}\, a)$ where $a \in ME_{\langle s,\sigma\rangle}$ for some $\sigma \in \mathsf{Types}$, then $[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![(\check{}\, a)]\!]^{\mathcal{M},w,t,h} = [\![a]\!]^{\mathcal{M},w,t,h}(w,t)$

The definition of truth is similar to that for L_1:

Definition 4.5 (Truth of Formulae)**.**

1. We say that a formula $\varphi : t$ is true relative to a model $\mathcal{M}$, $(w,t) \in W \times T$ and an assignment g, if $[\![\varphi]\!]^{\mathcal{M},w,t,g} = 1$
2. We say that $\varphi : t$ is true relative to $\mathcal{M}$ and $(w,t) \in W \times T$, and we write $[\![\varphi]\!]^{\mathcal{M},w,t} = 1$, if for all assignments g, $[\![\varphi]\!]^{\mathcal{M},w,t,g} = 1$

Proposition 4.1. For every expression $\alpha \in ME$, the extension $[\![\alpha]\!]^{\mathcal{M},w,t,h}$ of α is uniquely defined.

Proof. The proof is by recursion, i.e., by structural induction, on $\alpha \in ME$. □

Definition 4.6 (Montague Intension)**.** For every expression $A \in ME$, the *Montague intension* $[\![A]\!]^{\mathcal{M},h}$ of A, for a given model $\mathcal{M}$ and an assignment h, is the function $F : W \times T \to D$, such that, for every $(w,t) \in W \times T$:

$$F(w,t) = [\![A]\!]^{\mathcal{M},w,t,h} \tag{85}$$

Corollary 4.6.1. For every $A : \tau$, model $\mathcal{M}$, assignment g, and $(w,t) \in W \times T$:

$$[\![A]\!]^{\mathcal{M},h}(w,t) = [\![A]\!]^{\mathcal{M},w,t,h} \tag{86a}$$

$$[\![A]\!]^{\mathcal{M},h} = [\![\hat{}A]\!]^{\mathcal{M},w,t,h} \tag{86b}$$

Proof. Follows from Definitions 4.4.(xvii)–4.6. □

The semantics presented above looks different from that for L_1. Let us clarify some of the ideas that we use:

1. Some of the logical constants are now evaluated with respect to the model and are thought of as defining functions, just like any other function constants of L_{IL}. This is non-standard and was not done in Montague [12]. I prefer this approach since it gives a clear answer of what the logical constants mean (i.e., what their semantic values are). Thus, when we render English into L_{IL}, we will be able to render conjunctions and negations into something that has a semantic value by itself and not only with respect to other expressions. Otherwise, we would not be able to answer what, e.g., "not" means by itself in negated sentences — we can only give what the compound expressions, like "Jacques does not smoke", "Serge does not sing", etc., mean. For more details on categorematic logical constants, see Westerståhl [18]. For syncategorematic logical constans, on the other hand, see Winter [20]

2. The λ symbol in Definition 4.4.(iii) is used to form a new expression which denotes a function on the corresponding variable. This will be explained in more detail in Section 4.3

3. The $\Box$ symbol in Definition 4.4.(xiii) is used to regard the truth value of a formula with respect to the possible worlds and the points of time in which it is true. It can be thought of as corresponding to "necessarily", i.e. "in all possible worlds and at all points in time"

4. The **F** and **P** in Definition 4.4.(xiv)–(xv) are symbols similar to $\Box$ that are used to regard the truth value of a formula with repsect to the points in time in which it is true. They can be though of as corresponding to the future and past tense in English language, respectively

5. The $\hat{}$ symbol, in Definition 4.4.(xvi), is used to introduce intesional contexts. Asking what the semantic value of $\hat{a}$ is, is roughly the same as asking what the semantic value of a is at each $(w, t) \in W \times T$

6. The $\check{}$ symbol, in Definition 4.4.(xvii), removes intensional functionality from expressions. It can be thought of as taking an intensional expression and evaluating it at a given $(w, t) \in W \times T$

Note that Definition 4.3 implies that if we are to express a predicate of higher arity than 1, we then do it by creating a string of unary functions. Since we want our new language to be an extension of L_1, we need to prove that predicates of higher arity than 1 can be expressed in this new language.

We do that by proving the following simplified lemma, for the special case of untyped functions of two arguments.

Theorem 4.1 (Currying / Schönfinkelisation Theorem for the Untyped Functions of 2 Arguments)**.** Let A, B, C be nonempty sets, and:

$$C^{A\times B} = \{\, g \mid g : A \times B \to C \,\} \tag{87a}$$
$$C^{B} = \{\, h \mid h : B \to C \,\} \tag{87b}$$
$$U = \{\, f \mid f : A \to C^{B} \,\} \tag{87c}$$

There is a bijection between U and $C^{A\times B}$.

Proof. We define $F : U \to C^{A\times B}$ as follows:

For every $f \in U$, we define:

$$F(f) = g, \quad \text{where } g \text{ is the function } g \in C^{A\times B}, \text{ such that} \tag{88a}$$
$$g(x, y) = f(x)(y), \quad \text{for every } x \in A,\ y \in B \tag{88b}$$

First, we need to show that the function F is well-defined. For this, we need to prove the following:

(1) The function g in (88a), such that it satisfies (88a), exists

(2) The function g in (88a) is unique.

(1) follows, because, for every $x \in A$, $y \in B$ the value $f(x)(y)$ is defined, by $f \in U$, where U is as in (87c).

To see that (2) is the case, assume that there are $g_1, g_2 \in C^{A \times B}$, $g_1 \neq g_2$, that are such that (88b) holds, i.e.: $g_1(x, y) = f(x)(y)$, for every $x \in A$, $y \in B$; and $g_2(x, y) = f(x)(y)$, for every $x \in A$, $y \in B$. Then $g_1(x, y) = g_2(x, y)$, for every $x \in A$, $y \in B$, and by this, $g_1 = g_2$. This contradicts the assumption $g_1 \neq g_2$. Thus, it is not true, and g is unique.

Second, we show that the function F is surjective. Let $m : A \times B \to C$. If we can construct a function $f \in U$, i.e., $f : A \to C^B$, such that $F(f) = m$ we are done. We define $f : A \to C^B$ as follows. For each $a \in A$, let $f(a) = f_a$, where $f_a : B \to C$, such that $f_a(y) = m(a, y)$, for all $y \in B$. We now have, $F(f)(x, y) = f(x)(y) = f_x(y) = m(x, y)$, for all $x \in A$, $y \in B$, and by this, $F(f) = m$. Therefore, F is surjective.

Third, we show that the function F defined by (88a)-(88b), is injective. Let $f_1, f_2 \in U$, with $f_1 \neq f_2$. We then have $f_1(a) \neq f_2(a)$ for some $a \in A$. But $f_1(a)$ and $f_2(a)$ are functions, such that $f_1(a) : B \to C$ and $f_2(a) : B \to C$. So $f_1(a) \neq f_2(a)$ only if $f_1(a)(b) \neq f_2(a)(b)$, for some $b \in B$. By (88a)-(88b), it follows that $F(f_1)(a, b) \neq F(f_2)(a, b)$, and thus, $F(f_1) \neq F(f_2)$. Therefore, F is injective. □

Corollary 4.1.1. For any basic type σ, there is a bijection between $D_{\langle \sigma \times \sigma, \sigma \rangle}$ and $D_{\langle \sigma, \langle \sigma, \sigma \rangle \rangle}$.

Corollary 4.1.2. (1) There is a bijection between $D_{\langle e \times e, e \rangle}$ and $D_{\langle e, \langle e, e \rangle \rangle}$
(2) There is a bijection between $D_{\langle e \times e, t \rangle}$ and $D_{\langle e, \langle e, t \rangle \rangle}$
(3) There is a bijection between $D_{\langle s \times e, t \rangle}$ and $D_{\langle s, \langle e, t \rangle \rangle}$.

The special case Theorem 4.1, can be generalised to the Currying / Schönfinkelisation Theorem for the typed functions of n-arguments, by double recursion. I do not give the proof here — only a brief sketch of how it can be done. The first step would be to define the type system and domains for the multi-argument functions. The second step is to define a translation from this type system to the types in Types. This is done by induction on type structure. The third step is to define a bijective translation — a Curry coding — of the multi-argument functions to the unary functions. This is done by induction on the multi-argument types. Then, the proof that currying is bijective is by induction.

Note that the order of the arguments in Theorem 4.1 does not agree with the order of arguments that we are using. That is, Theorem 4.1 shows that we can transform arguments on the form (α_1, α_2) to a sequence of two arguments on the form $(\alpha_1)(\alpha_2)$, whereas we prefer (for our purposes) the reverse order $(\alpha_2)(\alpha_1)$. This is, however, not a problem since we can easily define a bijection between $C^{A \times B}$ and $C^{B \times A}$. Consequently, we are free to choose which order to use. In fact, for a generalised version of the theorem, we can establish a bijection between $C^{A_1 \times \cdots \times A_n}$ and $C^{A_{\delta(1)} \times \cdots \times A_{\delta(n)}}$ for any permutation δ on $\{1, \ldots, n\}$ (where $n \in \mathbb{N}$) and collection of sets $\{A_1, \ldots A_n\}$. I omit the proof, but note that it is a consequence of the fact that δ, by definition, is a bijection.

4.3 Some Features of λ-Calculus

Before we explore the applications of L_{IL}, we explain how the λ symbol in Definition 4.4.(iii) is used. The system in which we will work with L_{IL} is a system of *typed lambda calculus*. We introduce two new operations in Definition 4.7:

Notation 16. Often, we use the following notation, for any $A, B \in \mathrm{ME}_\tau$

$$A = B \Longleftrightarrow [\![A]\!]^{\mathcal{M},h} = [\![B]\!]^{\mathcal{M},h}, \tag{89}$$

for every model $\mathcal{M}$, and variable assignment h.

Definition 4.7. Let $\gamma : \tau$, such that the expression $\alpha \in \mathsf{Const}_\sigma$ occurs $n \geq 0$ times in γ. Then, if $x \in \mathsf{Vars}_\sigma$ and x does not occur in γ, i.e., x is a fresh variable, let $\gamma\{\, \alpha :\equiv x \,\}$ be the result of replacing all occurrences of α in γ with x.

Thus, we have:

$$(\lambda x\, (\gamma\{\, \alpha :\equiv x \,\})(\alpha) = \gamma \tag{90}$$

where $\lambda x\, (\gamma\{\, \alpha :\equiv x \,\}) : \langle \sigma, \tau \rangle$, $\alpha \in \mathsf{Const}_\sigma$ and $x \in \mathsf{Vars}_\sigma$ is a fresh variable. When we go from γ to $(\lambda x\, (\gamma\{\, \alpha :\equiv x \,\})(\alpha)$, we say *abstract* over α in γ. We also allow for reversing the operation. When we do this, we say we *convert*:

$$(\lambda x\, (\gamma\{\, \alpha :\equiv x \,\}))(\alpha) = (\gamma\{\, \alpha :\equiv x \,\})\{\, x :\equiv \alpha \,\} \tag{91}$$

Definition 4.8. (Restricted β-Conversion)

$$(\lambda x\, \gamma)(\alpha) = \gamma\{\, x :\equiv \alpha \,\} \tag{92}$$

given that the following conditions are satisfied:

(1) A general restriction for β-conversion in λ-calculus:

- Every free occurrence of x in γ is such that it is not in the scope of λy, $\forall y$, $\exists y$, for some $y \in \mathsf{FreeV}(\alpha)$. If this is the case, we say that α *is free for (replacing) x in γ.*

(2) A restriction that is specific for L_{IL}. At least one of the following two conditions are satisfied:

(a) Every free occurrence of x in γ is such that it is not in the scope of $\hat{}$, $\Box$, $\mathbf{F}$, $\mathbf{P}$

(b) The expression α in question is constructed from variables and expressions that can be of the form $\hat{}A$ or $\Box A$, by using the logic constants $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, and λ, application, $\forall$, $\exists$, $=$ (i.e., without the down symbol $\check{}$ and constants). If this is the case, we say that α is *modally closed.*

The condition (2)b can be thought of as (and can be proved to be) saying that α does not depend on world-time pairs, i.e.:

$$[\![\alpha]\!]^{\mathcal{M},w,t,h} = [\![\alpha]\!]^{\mathcal{M},w',t',h} \tag{93}$$

for all $(w,t),(w',t') \in W \times T$.

Notation 17. When we apply β-conversion, we shall write $A \xrightarrow{\mathsf{convert}} B$.

So, the λ-symbol designates an abstraction operator for forming expressions denoting functions. About details for the restrictions, see Musken [14] and Dowty et al. [3].

Example 4.2. Let $x \in \mathsf{Vars}_e$, $\mathbf{j} \in \mathsf{Const}_e$, $P \in \mathsf{Vars}_{\langle e,t\rangle}$, $\mathbf{sing} \in \mathsf{Const}_{\langle e,t\rangle}$. We re-consider our previous rendering of "Jacques sings", now in L_{IL}. In (94a)–(94b), we have parentheses according to the syntax of λ-expressions of L_{IL}, given in Definition 4.1-(iii), by adding some extra parentheses, sometimes in different sizes, for visualisation clarity:

$$\mathbf{sing}(\mathbf{j}) \xrightarrow{\mathsf{abstract}} \Big[\lambda x \big(\mathbf{sing}(x)\big)\Big](\mathbf{j}) \tag{94a}$$

$$\xrightarrow{\mathsf{abstract}} \Big[\lambda P \Big(\lambda x \big(P(x)\big)\Big)(\mathbf{j})\Big](\mathbf{sing}) \tag{94b}$$

In the reverse direction, by β-conversion, we have (95a)–(95b):

$$\Big[\lambda P \Big(\lambda x \big(P(x)\big)\Big)(\mathbf{j})\Big](\mathbf{sing}) \xrightarrow{\mathsf{convert}} \Big(\lambda x \big(\mathbf{sing}(x)\big)\Big)(\mathbf{j}) \tag{95a}$$

$$\xrightarrow{\mathsf{convert}} \mathbf{sing}(\mathbf{j}) \tag{95b}$$

Lambda calculus is, of course, not only about applications to the semantic representation in the analysis of everyday human language analysis. It has many other applications. For example, the function constant f can be used to denote the intended function, by the equation $f(x) = x^2$. The same function can be denoted by the expression $\lambda x\,(x^2)$, without introducing the function constant f. Thus, if we are to calculate the value $f(2)$, we can use β-conversion, as in (96):

$$\big(\lambda x\,(x^2)\big)(2) = 2^2 \tag{96}$$

We finish this subsection by adding two more laws, which are closely related to abstraction and conversion in lambda calculus.

Definition 4.9 (α-Conversion)**.** Alphabetical variants by renaming bound variables:

Let $A \in \mathrm{ME}_\tau$, $x \in \mathsf{Vars}_\sigma$

$$\lambda x A = \lambda y\,(A\{\, x :\equiv y \,\}) \tag{97a}$$

given that y has no occurrences in A and that y is free for replacing x in A.

Proposition 4.2 (Restricted Substitution for L_{IL})**.** For any $x \in \mathsf{Vars}_\tau$, $A, B \in \mathrm{ME}_\tau$ and $\gamma \in \mathrm{ME}_\sigma$ which satisfy the conditions (1) and (2)a or (2)b, the following holds:

$$[\![A = B]\!]^{\mathcal{M},h} = 1 \implies [\![\gamma\{\, x :\equiv A \,\}]\!]^{\mathcal{M},h} = [\![\gamma\{\, x :\equiv B \,\}]\!]^{\mathcal{M},h} \tag{98}$$

Substitution is restricted by using conditions similar to those in Definition 4.8 for A, B, x, γ.

(1) A, B are *free for (replacing) x in γ*.

(2) (a) Every free occurrence of x in γ is such that it is not in the scope of $^\wedge$, $\Box$, $\mathbf{F}$, $\mathbf{P}$

(b) The expressions A, B have the same intensions, by satisfying the following condition

$$^\wedge A = {}^\wedge B \tag{99}$$

Proof. Omitted. The proof is similar to (but longer and more complex than) the proof of Theorem 3.2. □

4.4 Down-Up Cancellation

We add one more rule. We note the following consequence of Definition 4.3.(xvi)–(xvii).

Proposition 4.3. For every expression A of L_{IL}, for every model $\mathcal{M}$, every variable assignment h for $\mathcal{M}$, every $w \in W$ and $t \in T$, the following holds:

$$[\![^\vee(^\wedge A)]\!]^{\mathcal{M},w,t,h} = [\![A]\!]^{\mathcal{M},w,t,h} \tag{100a}$$

$$[\![^\vee(^\wedge A)]\!]^{\mathcal{M},h} = [\![A]\!]^{\mathcal{M},h} \tag{100b}$$

Proof. Assume arbitrary choices of an expression A of L_{IL}, a model $\mathcal{M} = (D, W, T, \leq, I)$, an assignment h, $w \in W$ and $t \in T$. Then, by the definitions:

$$[\![^\vee(^\wedge A)]\!]^{\mathcal{M},w,t,h} = [\![^\wedge A]\!]^{\mathcal{M},w,t,h}(w,t) \tag{101a}$$

$$= g(w,t), \text{ where } g \text{ is the function such that } g(w',t') = [\![A]\!]^{\mathcal{M},w',t',h}, \text{ for all } (w',t') \in W \times T \tag{101b}$$

$$= [\![A]\!]^{\mathcal{M},w,t,h} \tag{101c}$$

This proves (100a). (100b) is an immediate consequence. □

Definition 4.10. For any expression A of L_{IL} we define the following operation:

$$(^\vee(^\wedge A)) \xrightarrow{\mathsf{cancel}} A \tag{102}$$

Note that the other order of cancellation does not hold. We show this by constructing a counterexample. Let $\mathcal{M} = (\{a, b\}, W, T, \leq, I)$ where $A : \langle s, \langle s, e\rangle\rangle$, $T = \{t\}$ and $W = \{w_1, w_2\}$. Assume that:

$$[\![\hat{}(\check{} A)]\!]^{\mathcal{M},w,t,h} = [\![A]\!]^{\mathcal{M},w,t,h} \tag{103}$$

Then, from (103), the following equality must hold:

$$[\![\hat{}(\check{} A)]\!]^{\mathcal{M},w,t,h}(w_1, t) = [\![A]\!]^{\mathcal{M},w,t,h}(w_1, t), \text{ for all } w \in W \tag{104}$$

By Definition 4.3.(xvi)–(xvii), we have:

$$[\![\hat{}(\check{} A)]\!]^{\mathcal{M},w,t,h}(w_1, t) = [\![\check{} A]\!]^{\mathcal{M},w_1,t,h} = [\![A]\!]^{\mathcal{M},w_1,t,h}(w_1, t) \tag{105}$$

And so, by (105) and (104), we have that the choice of w in the $[\![A]\!]^{\mathcal{M},w,t,h}$ is arbitrary. Hence (106) must hold:

$$[\![A]\!]^{\mathcal{M},w_2,t,h}(w_1, t) = [\![A]\!]^{\mathcal{M},w_1,t,h}(w_1, t) \tag{106}$$

But (106) does not always hold, in each world and time. To see this, let I be such that:

$$I(A)(w_1, t) = B \tag{107a}$$
$$B(w_1, t) = a \tag{107b}$$
$$I(A)(w_2, t) = C \tag{107c}$$
$$C(w_1, t) = b \tag{107d}$$

So, we have:

$$[\![A]\!]^{\mathcal{M},w_2,t,h}(w_1, t) = I(A)(w_2, t)(w_1, t) = b \tag{108a}$$
$$[\![A]\!]^{\mathcal{M},w_1,t,h}(w_1, t) = I(A)(w_1, t)(w_1, t) = a \tag{108b}$$

And so, by (108a)–(108b), we have that (106) fails.

4.5 L_{IL} in Grammar of Human Language

In this section, we introducde a Montague grammar for semantic represenation of HL. For this purpose, we render HL into L_{IL}. If A is some HL expression, $\alpha : \tau$ is some L_{IL} expression and $A \xrightarrow{\text{render}} \alpha$, then α can be used as a logical form of A which we can use for semantic representation of some interpretation of A.

4.5.1 Syntactic Categories and Basic Expressions of a Montagovian Grammar

We shall translate a fragment of English to L_{IL} expressions, by rules that use the syntax of each of these languages.

Syntactic Category	Type	Words
S	t	Jacques sings
T	$\langle\langle s, \langle e, t\rangle\rangle, t\rangle$	Jacques, he
N	$\langle e, t\rangle$	boy
IV	$\langle e, t\rangle$	sings
TV	$\langle\langle s, \langle\langle s, \langle e, t\rangle\rangle, t\rangle\rangle, \langle e, t\rangle\rangle$	is taller than
Det	$\langle\langle s, \langle e, t\rangle\rangle, \langle\langle s, \langle e, t\rangle\rangle, t\rangle\rangle$	the
P	$\langle\langle s, \langle\langle s, \langle e, t\rangle\rangle, t\rangle\rangle, \langle\langle s, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle\rangle$	under
Adj	$\langle\langle s, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle$	silly
Adv	$\langle\langle s, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle$	very
SAdv	$\langle\langle s, t\rangle, t\rangle$	necessarily
SCP	$\langle\langle s, t\rangle, \langle e, t\rangle\rangle$	thinks that
ICP	$\langle\langle s, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle$	attempt to
Conj	$\langle t, \langle t, t\rangle\rangle$	and
Neg	$\langle t, t\rangle$	not
Conj-S	$\langle t, t\rangle$	and Serge sings
Conj-IV	$\langle\langle e, t\rangle, \langle e, t\rangle\rangle$	and sings
Conj-Adj	$\langle\langle\langle s, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle, \langle\langle s, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle\rangle$	and silly
Conj-T	$\langle\langle\langle s, \langle e, t\rangle\rangle, t\rangle, \langle\langle s, \langle e, t\rangle\rangle, t\rangle\rangle$	and Jacques

Table 2: Assignment of L_{IL}-types to Syntactic Categories of HL in the Montagovian Grammar AGr

We begin by assigning types to each of the syntactic categories of human language we suggested in Table 1, now re-introduced, with some differences, in Table 2.

For syntactic categories of English language, similar to the ones in Table 2, and for similar type assignments, see Montague [12], Dowty et al. [3] and Loukanova [9].

Some things have changed in the transition from Table 1 to Table 2. First, the categories VP, PP and AdjP are missing in Table 2. The reason for this is that they have been conflated with IV, Adv and Adj, respectively. This is mainly for the sake of simplifying syntactic representation and it is something we could have chosen to do from the very start. In this essay, we do not provide sophisticated syntax of human language. Such simplifications would not have been suitable if we were interested in a very detailed syntax of HL. There are reasons to reserve IV for a lexical cateogory whilst leaving VP for more complex expressions. For details on formal syntax of English, see Sag et. al [16], Kim and Sells [8]. Here, we use L_{IL} to provide logic expressions for semantic representations.

Secondly, the syntactic category NP from Table 1 has been replaced by the syntactic category T in Table 2. Sometimes T is referred to as the syntactic category of *term phrases*. T can, for the purposes in this work, be seen as the same category as NP In future work, I wish to distinguish between the two. Montague [12] gives some of the expressions of category T a very differ-

ent interpretation than what is intuitive for some expressions of the syntactic category NP. For instance, sometimes, we would prefer to interpret some NPs as denoting individuals, e.g., when they have the grammatical form of countable nouns in singular form — hence, they should, intuitively, be of type e. Often in Montague Grammars, NPs, i.e. Ts, are interpreted as denoting sets of properties of individuals. Consequently, it makes sense to let NPs take VPs as arguments rather than vice versa. We shall see how this is used to solve the problems of the quantifiers covered previously. I emphasise that was one of Montague's major novelties at the time, which continues to be used in Montagovian grammars. The type of a property is $\langle s, \langle e, t\rangle\rangle$, and thus, the type of a set of properties is $\langle\langle s, \langle e, t\rangle\rangle, t\rangle$. A proper name, like, "Serge" refers[11] to the set of all properties that the individual Serge satisfies — e.g. properties like {is a singer, smokes, is French, ... }.

As a consequence of the above, I have also replaced Conj-VP, Conj-AdjP and Conj-NP with Conj-IV, Conj-Adj and Conj-T respectively.

For each syntactic category C, Montague [12] introduced $\mathrm{B_C}$ — the set of the basic expressions of the category C. $\mathrm{B_C}$ are nonempty for the categories C that have basic expressions, i.e., words in the considered fragment of English. We define $\mathrm{B_C}$ for our fragment Γ of English:

$$\mathrm{B_T} = \{\,\mathrm{Serge}, \mathrm{Jacques}, \mathrm{he}_0, \mathrm{she}_0, \mathrm{it}_0, \ldots, \mathrm{he}_n, \mathrm{she}_n, \mathrm{it}_n, \ldots\} \tag{109}$$

$$\mathrm{B_N} = \{\text{cat, boy}\} \tag{110}$$

$$\mathrm{B_{IV}} = \{\text{sing(s), smoke(s)}\} \tag{111}$$

$$\mathrm{B_{TV}} = \{\text{is taller than, writes, }\} \tag{112}$$

$$\mathrm{B_{Det}} = \{\text{some, every, the}\} \tag{113}$$

$$\mathrm{B_P} = \{\text{by}\} \tag{114}$$

$$\mathrm{B_{Adv}} = \{\text{well, rapidly}\} \tag{115}$$

$$\mathrm{B_{Adj}} = \{\text{silly, blue}\} \tag{116}$$

$$\mathrm{B_{SAdv}} = \{\text{necessarily}\} \tag{117}$$

$$\mathrm{B_{SCP}} = \{\text{thinks that}\} \tag{118}$$

$$\mathrm{B_{ICP}} = \{\text{attempt to}\} \tag{119}$$

$$\mathrm{B_{Conj}} = \{\text{and, or}\} \tag{120}$$

$$\mathrm{B_{Neg}} = \{\text{does not}\} \tag{121}$$

$$\mathrm{B_C} = \varnothing \quad \text{if } C \notin \{\text{T, N, IV, TV, Det, P, Adv, Adj, SAdv, SCP, ICP, Conj, Neg}\} \tag{122}$$

We will also define $\mathrm{P_C}$ for each category C as the set of all phrases of that category, see (P1)–(P26).

[11] This is, of course, an assumption that comes with a great deal of philosophical implications, but we shall not dwell on this here, because we concentrate on a mathematical representation of the style of PTQ, see Montague [12].

4.5.2 Type-Lift Rules

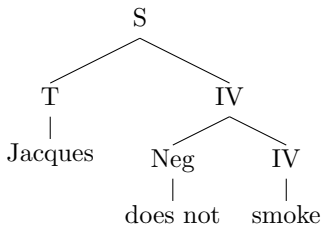
There are some problems with using L_{IL} for semantic representations in a grammar of human language. Some of the types do not seem to match.

Consider the sentences (A)–(C), and their plausible renderings. Note that the objects rendering the proper names Jacques and Serge are treated differently from the L_1-renderings. This is a consequence of their syntactic category of terms T being associated with the type $\langle\langle s, \langle e, t\rangle\rangle, t\rangle$ rather than e. The proper nouns Jacques and Serge are rendered to the constants **J** and **S**, correspondingly. This results in another mismatch between rendering the sentences (A)–(C), and, in addition, non-well-formed expressions ($\neg$**smoke**), (**smoke**$\wedge$**sing**), $\wedge$(**sing**)(**smoke**), $\wedge$(**sing**), (**J**$\wedge$**S**), $\wedge$(**S**)(**J**), $\wedge$(**S**). Thus, these candidate-renderings are incorrect. We explain some of the problems below. Note that the trees in (123)–(127) use the new syntactic categories in Table 2.

Note that the trees in (123), (125) and (127) represent renderings of the sentences (A)–(C), where the coordinated expressions are generated by adding their components one-by-one. The leaves are English words, not constants in any logic language. We use auxiliary syntactic categories as this will able us to render negation and coordination words into logical constants (of corresponding Curry type) in a categorematic way.

(A) Jacques does not smoke

A possible syntactic analysis of this sentence is given in (123):



(123)

However, if we are to let

$$\text{not} \xrightarrow{\text{render}} \neg$$

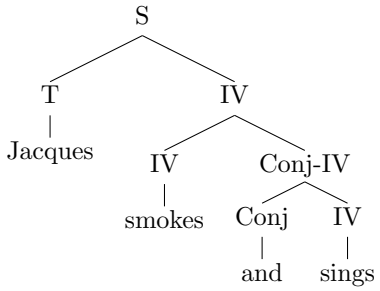
we will end up with non-well-formed expressions. We want to be able to reflect that "does not smoke" belongs to the category IV, just like "smoke". However, if we are to follow the phrase structure given in (123), we appear to get 124, which is not well-formed, primarily due to the fact that **smoke** is not of type t and therefore cannot be negated.

$$\text{* incorrect *} \qquad \text{Jacques does not smoke} \xrightarrow{\text{render}} \mathbf{J}(\neg\mathbf{smoke}) \tag{124}$$

(B) Jacques smokes and sings

A possible syntactic analysis of this sentence is given in (125):

(125)



If we are to let

$$\text{and} \xrightarrow{\text{render}} \wedge$$

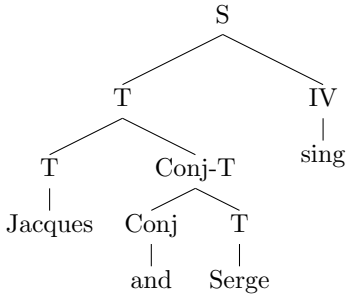
it is unclear how to use this to combine the verbs "smokes" and "sings", since $\wedge : \langle t, \langle t, t\rangle\rangle$ and any rendering of the two verbs should be of type $\langle e, t\rangle$. Also, if this coordination would result in something of type t, there is no way for us to combine this with the expression rendered by "Jacques", which, clearly, is not of type $\langle t, t\rangle$. Hence, the redering in (126) which appears to be the rendering reflected in (125) does not make sense.

$$\text{* incorrect *} \quad \text{Jacques smokes and sings} \xrightarrow{\text{render}} \mathbf{J}(\mathbf{smoke} \wedge \mathbf{sing}) \quad (126)$$

(C) Jacques and Serge sing

A possible syntactic analysis of this sentence is given in (127):

(127)



Again, if we are to let

$$\text{and} \xrightarrow{\text{render}} \wedge$$

it is unclear how to use this to combine the term phrases "Jacques" and "Serge", since $\wedge : \langle t, \langle t, t\rangle\rangle$ and any render of a term phrase should be of type $\langle\langle s, \langle e, t\rangle\rangle, t\rangle$. As before, we also cannot let the coordinated term phrase be of type t, as this would prevent us from combining it with the expression rendered by "sing". Hence, the rendering in (128) which appears to be the rendering reflected in (127) does not make sense.

$$\text{* incorrect *} \qquad \text{Jacques and Serge sing} \xrightarrow{\mathsf{render}} (\mathbf{J} \wedge \mathbf{S})(\mathbf{sing}) \quad (128)$$

Let us summarise the problems above. In (123), we incorrectly let $\neg : \langle t, t\rangle$ take **sing** as an argument which results in an expression that we would like to be of type $\langle e, t\rangle$. But **sing** is of type $\langle e, t\rangle$. So $\neg$ behaves as if it were of type $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$.

In (125), we have the problem that $\wedge : \langle t, \langle t, t\rangle\rangle$ somehow takes $\mathbf{sing} : \langle e, t\rangle$ as an argument to then output an expression that takes $\mathbf{smoke} : \langle e, t\rangle$ as an argument, which, in turn, outputs another expression of type $\langle e, t\rangle$. So $\wedge$ behaves as if it were of type $\langle\langle e, t\rangle, \langle\langle e, t\rangle, \langle e, t\rangle\rangle\rangle$ rather than $\wedge : \langle t, \langle t, t\rangle\rangle$.

In (127), the symbol $\wedge$ is misused in a similar way, but as if it were for a conjunction of individual terms, i.e., of two expressions of type $\langle\langle s, \langle e, t\rangle\rangle, t\rangle$, instead of conjunction of formulae. Hence, $\wedge$ has been misused, as if it were of type $\langle\langle\langle s, \langle e, t\rangle\rangle, t\rangle, \langle\langle\langle s, \langle e, t\rangle\rangle, t\rangle, \langle\langle s, \langle e, t\rangle\rangle, t\rangle\rangle\rangle$.

There are different ways of solving this problem. My solution of choice, for this essay, is to introduce two type-shift rules. These were not included in Montague [12], but are useful if we are to interpret coordination and negation words like "and", "not", and the corresponding logic connectives $\wedge$, $\neg$, categorematically. That is, we would have preferred to place coordination (conjunction, disjunction) and negation words in a syntactic category that is appropriate for them, by giving them corresponding interpretations, similarly to any other words.

In Montague [12], on the other hand, we are given a syncategorematic interpretations of "and", "not", etc., and of the corresponding connectives. Hence, the logical constants are not assigned types.

The following definitions are based on Westerståhl [19]:

Definition 4.11. Let $T : \langle \tau, \tau\rangle$ and $\sigma, \tau \in \mathsf{Types}$. The operation $\mathsf{geach}_1(T)$ is defined by (129b)–(129a):

$$\mathsf{geach}_1(T) = (\lambda P\, (\lambda x\, T(P(x)))) \quad (129a)$$

$$\mathsf{geach}_1(T) : \langle\langle \sigma, \tau\rangle, \langle \sigma, \tau\rangle\rangle \quad (129b)$$

where $P : \langle \sigma, \tau\rangle$ and $x : \sigma$. For this essay, we use the following special cases (130a)–(130b) of geach_1 from (129b)–(129a):

$$\mathsf{geach}_1^a(T) : \langle\langle \sigma, \tau\rangle, \langle \sigma, \tau\rangle\rangle \text{ with } \sigma = e \quad (130a)$$

$$\mathsf{geach}_1^b(T) : \langle\langle \sigma, \tau\rangle, \langle \sigma, \tau\rangle\rangle \text{ with } \sigma = \langle s, \langle e, t\rangle\rangle \quad (130b)$$

So, if $T = \neg$, with $\tau = t$, we have (131):

$$\mathsf{geach}_1^a(\neg) : \langle\langle e, t\rangle, \langle e, t\rangle\rangle \quad (131)$$

The operation (i.e., operator) geach_1^a, in (131) is useful for representing negated intransitive verbs, of the syntactic category IV, which we show in (135a)–(135b).

The operation geach_1^b will be used later in the essay to handle the negation of adjectives.

Definition 4.12. Let $T : \langle \tau, \langle \tau, \tau \rangle \rangle$ and $\sigma, \tau \in \mathsf{Types}$. The operation geach_2 is defined by (132a)–(132b):

$$\mathsf{geach}_2(T) = (\lambda P\,(\lambda Q\,(\lambda x\,(T(P(x))(Q(x))))))) \tag{132a}$$

$$\mathsf{geach}_2(T) : \langle\langle \sigma, \tau \rangle, \langle\langle \sigma, \tau \rangle, \langle \sigma, \tau \rangle\rangle\rangle \tag{132b}$$

where $P, Q : \langle \sigma, \tau \rangle$ and $x : \sigma$. For this essay we use the following special cases (133a)–(133b) of geach_2 from (132a)–(132b):

$$\mathsf{geach}_2^a(T) : \langle\langle \sigma, \tau \rangle, \langle\langle \sigma, \tau \rangle, \langle \sigma, \tau \rangle\rangle\rangle \text{ with } \sigma = e \tag{133a}$$

$$\mathsf{geach}_2^b(T) : \langle\langle \sigma, \tau \rangle, \langle\langle \sigma, \tau \rangle, \langle \sigma, \tau \rangle\rangle\rangle \text{ with } \sigma = \langle s, \langle e, t \rangle\rangle \tag{133b}$$

Thus, for $T = \wedge$, with $T : \langle t, \langle t, t \rangle\rangle$, we have (134a)–(134b):

$$\begin{array}{l} \mathsf{geach}_2^a(\wedge) : \langle\langle e, t \rangle, \langle\langle e, t \rangle, \langle e, t \rangle\rangle\rangle \\ \text{for } T = \wedge \text{ with } \tau = t,\ \sigma = e \end{array} \tag{134a}$$

$$\begin{array}{l} \mathsf{geach}_2^b(\wedge) : \langle\langle\langle s, \langle e, t \rangle\rangle, t \rangle, \langle\langle\langle s, \langle e, t \rangle\rangle, t \rangle, \langle\langle s, \langle e, t \rangle\rangle, t \rangle\rangle\rangle \\ \text{for } T = \wedge \text{ with } \tau = t,\ \sigma = \langle s, \langle e, t \rangle\rangle \end{array} \tag{134b}$$

We shall show how $\mathsf{geach}_2^a(\wedge)$ can be used in rendering of coordinated expressions of the syntactic category IV of intransitive verb phrases, and $\mathsf{geach}_2^b(\wedge)$ for coordinated *term phrases* of category T, by the conjunction connector "and".

We now show how Definitions 4.11–4.12 may be used towards solutions of the problems in (123)–(127). In what follows, recall that $\mathbf{smoke}, \mathbf{sing} : \langle e, t \rangle$ and $\mathbf{J}, \mathbf{S} : \langle\langle s, \langle e, t \rangle\rangle, t \rangle$.

Towards a Solution of the Problems in (123)**.** If we let:

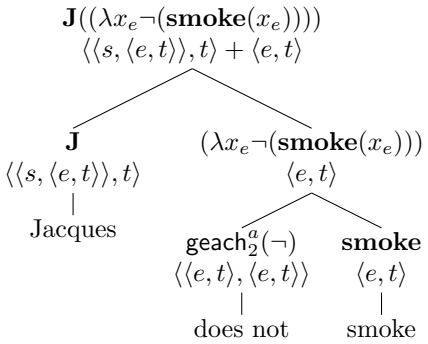
$$\text{does not} \xrightarrow{\mathsf{render}} \mathsf{geach}_1^a(\neg)$$

rather than $\neg$, we can make sense of the rendering suggested by the tree in (123). We get:

$$\mathsf{geach}_1^a(\neg)(\mathbf{smoke}) = \Big(\lambda P_{\langle e,t \rangle}(\lambda x_e\, \neg(P_{\langle e,t \rangle}(x_e)))\Big)(\mathbf{smoke}) \ : \ \langle e, t \rangle \tag{135a}$$

$$\xrightarrow{\mathsf{convert}} (\lambda x_e\, \neg(\mathbf{smoke}(x_e))) \ : \ \langle e, t \rangle \tag{135b}$$

Using (135a)–(135b), we give the following preliminary rendering of "Jacques does not smoke" in (136):



(136)

The idea is that the L_{IL} expression in top node should simplify too, but this cannot occur yet due to the remaining problem of type mismatch between **J** and the new candidates for negation of HL expressions of syntactic category IV, given in (135a)–(135b). We shall see that this can be taken care of by using the ˆ and ˇ symbols. We will return to this problem later.

Towards a Solution of the Problems in (125)**.** If we let:

$$\text{and} \xrightarrow{\text{render}} \mathsf{geach}_2^a(\wedge)$$

rather than $\wedge$, we are able to make sense of the tree structure given in (125). We get:

$$\mathsf{geach}_2^a(\wedge)(\mathbf{sing}) = \tag{137a}$$

$$\Big(\lambda P_{\langle e,t\rangle}\,(\lambda Q_{\langle e,t\rangle}\,(\lambda x_e\,(\wedge(P_{\langle e,t\rangle}(x_e))(Q_{\langle e,t\rangle}(x_e))))))\Big)(\mathbf{sing}) \tag{137b}$$

$$\xrightarrow{\text{convert}}\ (\lambda Q_{\langle e,t\rangle}\,(\lambda x_e\,(\wedge(\mathbf{sing}(x_e))(Q_{\langle e,t\rangle}(x_e))))) \ :\ \langle\langle e,t\rangle,\langle e,t\rangle\rangle \tag{137c}$$

and hence:

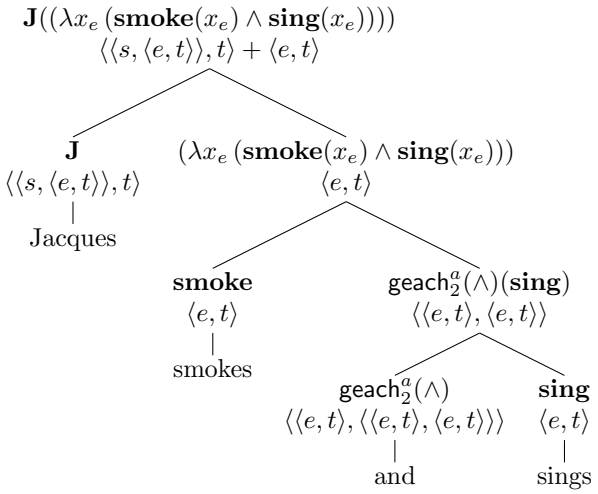
$$\Big[\mathsf{geach}_2^a(\wedge)(\mathbf{sing})\Big](\mathbf{smoke}) \tag{138a}$$

$$\xrightarrow{\text{convert}}\ \Big(\lambda Q_{\langle e,t\rangle}\,(\lambda x_e\,(\wedge(\mathbf{sing}(x_e))(Q_{\langle e,t\rangle}(x_e))))\Big)(\mathbf{smoke}) \tag{138b}$$

$$\xrightarrow{\text{convert}}\ (\lambda x_e\,(\wedge(\mathbf{sing}(x_e))(\mathbf{smoke}(x_e)))) = \tag{138c}$$

$$(\lambda x_e\,(\mathbf{smoke}(x_e) \wedge \mathbf{sing}(x_e))) \ :\ \langle e,t\rangle \tag{138d}$$

And so a preliminary rendering of "Jacques smokes and sings" is that given by the tree in (139):



(139)

As before, the idea is that the expression in the top node should simplify to $\mathbf{smoke}(\mathbf{j}_e) \wedge \mathbf{sing}(\mathbf{j}_e)$, which will be possible when we have solved the problem with the type mismatch.

Towards a Solution of the Problems in (127)**.** If we let:

$$\wedge \xrightarrow{\mathsf{render}} \mathsf{geach}_2^b(\wedge)$$

rather than $\wedge$, we can make sense of the tree structure in (127). For readability, let $\sigma = \langle s,\langle e,t\rangle\rangle$. We get:

$$\mathsf{geach}_2^b(\wedge)(\mathbf{S}) = \tag{140a}$$

$$\Big(\lambda P_{\langle\sigma,t\rangle}\,(\lambda Q_{\langle\sigma,t\rangle}\,(\lambda x_\sigma(\wedge(P_{\langle\sigma,t\rangle}(x_\sigma))(Q_{\langle\sigma,t\rangle}(x_\sigma))))))\Big)(\mathbf{S}) \tag{140b}$$

$$\xrightarrow{\mathsf{convert}} \big(\lambda Q_{\langle\sigma,t\rangle}\,(\lambda x_\sigma\,(\wedge(\mathbf{S}(x_\sigma))(Q_{\langle\sigma,t\rangle}(x_\sigma))))\big)\ :\ \langle\langle\sigma,t\rangle,\langle\sigma,t\rangle\rangle \tag{140c}$$

Hence:

$$\Big[\mathsf{geach}_2^b(\wedge)(\mathbf{S})\Big](\mathbf{J}) = \tag{141a}$$

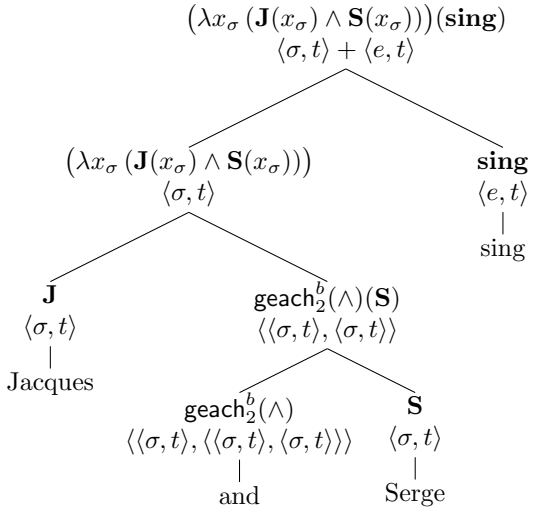
$$\xrightarrow{\mathsf{convert}} \Big[\lambda Q_{\langle\sigma,t\rangle}\,(\lambda x_\sigma\,(\wedge(\mathbf{S}(x_\sigma))(Q(x_\sigma))))\Big](\mathbf{J}) \tag{141b}$$

$$\xrightarrow{\mathsf{convert}} (\lambda x_\sigma\,(\wedge(\mathbf{S}(x_\sigma))(\mathbf{J}(x_\sigma)))) = \tag{141c}$$

$$(\lambda x_\sigma\,(\mathbf{J}(x_\sigma) \wedge \mathbf{S}(x_\sigma)))\ :\ \langle\sigma,t\rangle \tag{141d}$$

And so a preliminary rendering of "Jacques and Serge sing" is that given by

the tree in (142):



(142)

As before, the idea is that the expression in the top node should simplify, which, yet again, will be possible when we have solved the problem with the type mismatch.

4.6 Montagovian Grammar AGr

We have now covered preliminaries needed to present a Montagovian Grammar AGr, which is very similar to the original PTQ grammar introduced by Montague [12].

4.6.1 Syntax of a Fragment of English

Syntactic Categories of AGr Similarly to PTQ in Montague [12], our grammar AGr has syntactic categories, which are listed in Table 2, along with a mapping of the categories to corresponding L_{IL} types.

Table 2 defines the set $\mathsf{SynCats}$ of our syntactic categories of our fragment of English language, along with some basic expressions.

Basic Expressions of AGr The set of the basic expressions, for each syntactic category $\mathrm{C} \in \mathsf{SynCats}$, is given in (109), possibly with some more words, but without adding words that can diverge from our syntactic rules given below.

Syntax of English Expressions in AGr Now, we give the structural rules of our Montagovian grammar AGr for compound expressions of our fragment of English. The rules associate each expression with a syntactic category, i.e., as a member of a set $\mathrm{P_C}$ of expressions of category C.. We shall see that they either correspond to or extend the previously suggested phrase structure rules

(9a)–(9u). For similar grammars, see Dowty et al. [3] and Montague [12]. I use these grammars as the foundation of the grammar presented here, although I add rules for the auxiliary syntactic categories Conj-S, Conj-T etc., which are not in either of the cited texts.

Now, for each category C, we define the set $\mathrm{P_C}$ of expressions of category C:

(P1) If C is a syntactic category, then $\mathrm{B_C} \subseteq \mathrm{P_C}$

(P2) If $\delta \in \mathrm{P_N}$, then $F_0(\delta), F_1(\delta), F_2(\delta) \in \mathrm{P_T}$, where:

$$F_0(\delta) = \text{every } \delta \tag{143}$$

$$F_1(\delta) = \text{the } \delta \tag{144}$$

$$F_2(\delta) = \begin{cases} \text{a } \delta, & \text{if } \delta \text{ begins with a consonant (sound);} \\ \text{an } \delta, & \text{if } \delta \text{ begins with a vowel (sound)} \end{cases} \tag{145}$$

Note that all words in $\mathrm{B_N}$ in (110) and $\mathrm{B_{Adj}}$ in (116) begin with consonants, so we are in fact only interested in the first case of F_2.

(P3) If $\delta \in \mathrm{P_N}$ and $\alpha \in \mathrm{P_S}$, then:

$$F_{3,n}(\delta, \alpha) = \delta \text{ such that } \alpha' \tag{146}$$

where α' is just like α but with each occurrence of he_n and him_n replaced by he/she/it or him/her/it respectively, depending on the gender of the first $\mathrm{B_N}$-expression in δ

(P4) If $\delta \in \mathrm{P_T}$, $\alpha \in \mathrm{P_{IV}}$, then $F_4(\delta, \alpha) \in \mathrm{P_S}$ where:

$$F_4(\delta, \alpha) = \begin{cases} \delta\,\alpha^1 & \text{if } \delta \text{ is not coordinated with “and”;} \\ \delta\,\alpha^2 & \text{if } \delta \text{ is coordinated with “and”} \end{cases} \tag{147}$$

where α^1 is just like α but with its first verb replaced by its third-person, singular present version and α^2 is just like α but with its first verb replaced by its third-person, plural present version

(P5) If $\delta \in \mathrm{P_{TV}}$ and $\alpha \in \mathrm{P_T}$, then $F_5(\delta, \alpha) \in \mathrm{P_{IV}}$ where:

$$F_5(\delta, \alpha) = \begin{cases} \delta\,\alpha & \text{if } \alpha \neq \mathrm{he}_n \\ \delta\,\mathrm{him}_n & \text{otherwise} \end{cases} \tag{148}$$

(P6) If $\delta \in \mathrm{P_P}$ and $\alpha \in \mathrm{P_T}$, then $F_5(\delta, \alpha) \in \mathrm{P_{Adv}}$ (where F_5 is defined in (148))

(P7) If $\delta \in \mathrm{P_{SCP}}$ and $\alpha \in \mathrm{P_S}$, then $F_6(\delta, \alpha) = \delta\,\alpha \in \mathrm{P_{IV}}$

(P8) If $\delta \in \mathrm{P_{ICP}}$ and $\alpha \in \mathrm{P_{IV}}$, then $F_6(\delta, \alpha) = \delta\,\alpha \in \mathrm{P_{IV}}$

(P9) If $\delta \in \mathrm{P_{SAdv}}$ and $\alpha \in \mathrm{P_S}$, then $F_6(\delta, \alpha) = \delta\,\alpha \in \mathrm{P_S}$

(P10) If $\delta \in \mathrm{P_{Adj}}$ and $\alpha \in \mathrm{P_N}$, then $F_6(\delta, \alpha) = \delta\,\alpha \in \mathrm{P_N}$

(P11) If $\delta \in \mathrm{P_{Adv}}$ and $\alpha \in \mathrm{P_{IV}}$, then $F_7(\delta, \alpha) = \alpha\,\delta \in \mathrm{P_{IV}}$

(P12) If $\delta \in \mathrm{P_{Conj}}$ and $\alpha \in \mathrm{P_S}$, then $F_8(\delta, \alpha) = \delta\,\alpha \in \mathrm{P_{Conj\text{-}S}}$

(P13) If $\delta \in \mathrm{P_{Conj}}$ and $\alpha \in \mathrm{P_{IV}}$, then $F_8(\delta, \alpha) = \delta\,\alpha \in \mathrm{P_{Conj\text{-}IV}}$

(P14) If $\delta \in \mathrm{P_{Conj}}$ and $\alpha \in \mathrm{P_T}$, then $F_8(\delta, \alpha) = \delta\,\alpha \in \mathrm{P_{Conj\text{-}T}}$

(P15) If $\delta \in \mathrm{P_{Conj}}$ and $\alpha \in \mathrm{P_{Adj}}$, then $F_8(\delta, \alpha) = \delta\,\alpha \in \mathrm{P_{Conj\text{-}Adj}}$

(P16) If $\delta \in \mathrm{P_S}$ and $\alpha \in \mathrm{P_{Conj\text{-}S}}$, then $F_9(\delta, \alpha) = \delta\,\alpha \in \mathrm{P_S}$

(P17) If $\delta \in \mathrm{P_{IV}}$ and $\alpha \in \mathrm{P_{Conj\text{-}IV}}$, then $F_9(\delta, \alpha) = \delta\,\alpha \in \mathrm{P_{IV}}$

(P18) If $\delta \in \mathrm{P_T}$ and $\alpha \in \mathrm{P_{Conj\text{-}T}}$, then $F_9(\delta, \alpha) = \delta\,\alpha \in \mathrm{P_T}$

(P19) If $\delta \in \mathrm{P_{Adj}}$ and $\alpha \in \mathrm{P_{Conj\text{-}Adj}}$, then $F_9(\delta, \alpha) = \delta\,\alpha \in \mathrm{P_{Adj}}$

(P20) (*Quantification Rule*) If $\delta \in \mathrm{P_T}$ and $\alpha \in \mathrm{P_S}$, then $F_{10,n}(\delta, \alpha) \in \mathrm{P_S}$, where the syntactic operation $F_{10,n}$ is defined as follows, by (149):

$$F_{10,n}(\delta, \alpha) = \begin{cases} \alpha^1 & \text{if } \delta \neq \mathrm{he}_k \\ \alpha^2 & \delta = \mathrm{he}_k \end{cases} \tag{149}$$

where:

(1) α^1 is obtained from α by replacing, the first occurrence of he_n or him_n (the one that comes first) by δ, and all other occurrences of he_n and him_n, correspondingly by:

(a) he/she/it or him/her/it, with respect to the gender of δ (which, in this AGr, is the first T- or N-expression in δ), if δ is not a coordinated T-expression with the conjunction "and"

(b) they or them, if δ is a coordinated T with the conjunction"and"

(2) α^2 is the result of replacing, in α, all occurrences of $\mathrm{he}_n/\mathrm{him}_n$ by $\mathrm{he}_k/\mathrm{him}_k$

(P21) (*Quantification Rule*) If $\delta \in \mathrm{P_T}$ and $\alpha \in \mathrm{P_N}$, then $F_{10,n}(\delta, \alpha) \in \mathrm{P_N}$, where $F_{10,n}(\delta, \alpha)$ is as in (149)

(P22) (*Quantification Rule*) If $\delta \in \mathrm{P_T}$ and $\alpha \in \mathrm{P_{IV}}$, then $F_{10,n}(\delta, \alpha) \in \mathrm{P_{IV}}$, where $F_{10,n}(\delta, \alpha)$ is as in (149)

(P23) (*Future and Past Tenses*) If $\alpha \in \mathrm{P_T}$ and $\delta \in \mathrm{P_{IV}}$, then $F_i(\alpha, \delta) \in \mathrm{P_S}$ for $i = 11, 12$ where:

(1) $F_{11}(\alpha, \delta) = \alpha\,\delta'$ where δ' is the result of replacing the first verb in δ by its third person singular future tense

(2) $F_{12}(\alpha, \delta) = \alpha\,\delta''$ where δ'' is the result of replacing the first verb in δ by its third person singular past tense

(P24) If $\alpha \in \mathrm{P}_{\mathrm{Neg}}$ and $\delta \in \mathrm{P}_{\mathrm{IV}}$, then $F_{13}(\alpha, \delta) = \alpha\,\delta \in \mathrm{P}_{\mathrm{IV}}$

(P25) If $\alpha \in \mathrm{P}_{\mathrm{Neg}}$ and $\delta \in \mathrm{P}_{\mathrm{Adj}}$, then $F_{13}(\alpha, \delta) = \alpha\,\delta \in \mathrm{P}_{\mathrm{Adj}}$

(P26) If $\alpha \in \mathrm{P}_{\mathrm{Neg}}$ and $\varphi \in \mathrm{P}_{\mathrm{S}}$, then $F_{13}(\alpha, \delta) = \alpha\,\varphi \in \mathrm{P}_{\mathrm{S}}$

Some of these rules are not as straightforward as the phrase structure rules (9a)–(9u). Here follows an explanation of the rules

Explanation of (P1)–(P26)

1. (P1) simply claims that all basic expressions of some category C are to be part of all phrases of category C

2. (P2) defines ways to attach determiners to nouns. So, for instance:

$$F_1(\text{boy}) = \text{the boy} \tag{150}$$

3. (P3) and $F_{3,n}$ can be used to construct complex nouns that are simple relative clauses. For instance:

$$F_{3,n}(\text{boy}, \text{he}_n \text{ walks}) = \text{boy such that he walks} \tag{151}$$

4. (P4) combines a verb and a term phrase to make a sentence. It also makes sure that the verb agrees with the term phrase grammatically so that we do not end up with ungrammatical sentences such as "Serge and Jacques sings" or "Serge sing"

5. (P5) combines a transitive verb with a term phrase to make an intransitive verb. It also replaces he_n with him_n when needed

6. (P6) combines a preposition, of the syntactic category P, and a term phrase, of category T, to construct a prepositional phrase, PP. However, in with our syntactic categories here, we have conflated PP with Adv (since many prepositional phrases are used as adverbial expressions). This was also done in Montague [12]. So:

$$F_5(\text{by}, \text{the boy}) = \text{by the boy} \tag{152}$$

   The expression in (152) can be used as an adverbial — we can combine it with a verb phrase to form another (more complex) verb phrase

7. (P7)–(P11) are straightforward, corresponding clearly to rules in (9a)–(9u)

8. (P12)–(P19) are coordination rules, for constructing expressions with conjunct words, i.e., "and" and "or" in this work, and correspond to those in (9a)–(9u)

9. (P20) is an entirely new rule. It lets us *quantify* term phrases *into* sentences. That is, it lets us replace every occurrence of he_n/him_n by the term phrase in question. So, for example:

$$F_{10,n}(\text{Jacques}, \text{he}_n \text{ sings}) = \text{Jacques sings} \tag{153}$$

(P21) and (P22) work in the same way as (P20), but for nouns and intransitive verbs, respectively. These rules are important in Montagovian Grammars. They can be used to impose extra-syntax on English, for representation of de re and de dicto readings (recall the two readings of S2). Now, A2* can be treated as being represented by the "sentence":

S1** Some rational number, every integer is greater than it_0

10. (P23) are simple tense rules to deal with future and past tense:

$$F_{11}(\text{Jacques}, \text{sings}) = \text{Jacques will sing} \tag{154}$$

11. (P24)–(P26) are basic negation rules. We get some overgeneration, since, for example, (P24) gives rise to expressions like "Jacques does not sings". They also appear to give rise to expressions like "Does not Jacques sings" and "The does not blue man". These rules are not ideal, but we leave their improvement for future work.

Further, we note that in our AGr, we have represented the CFG with the phrasal rules (9a)–(9u). This is shown in Table 3.

4.6.2 Compositional Rendering of English into L_{IL}

We now introduce Montagovian rules for rendering our fragment of English language, generated by our grammar AGr, into L_{IL}.

Recall that the sets B_C of basic expressions of category C, are given in (109). Recall also that Table 2 defines a set $\mathsf{SynCats}$ of syntactic categories of our fragment of English language and a mapping Type from the categories to types of L_{IL}:

$$\mathsf{Type}\colon \mathsf{SynCats} \to \mathsf{Types} \tag{155}$$

Recall the brace notation in (79) and (80).

p.52, Notation 15.

Notation 18. For each $\mathbf{a} \in \mathsf{Const}_e$, we use the following abbreviation(s) (156):

$$\mathbf{a}^* = \big[\lambda P\left(\left(\check{}P\right)(\mathbf{a})\right)\big] = [\lambda P\left(\check{}P\right)(\mathbf{a})] = \lambda P\left[P\{\mathbf{a}\}\right] \tag{156}$$

for any $P \in \mathsf{Vars}_{\langle s,\langle e,t\rangle\rangle}$. We shall also use the abstract-operation, as in (157), to recover from the abbreviation:

$$\mathbf{a}^* \xrightarrow{\mathsf{abstract}} \big[\lambda P\left(\left(\check{}P\right)(\mathbf{a})\right)\big] \tag{157}$$

Similarly to Montague [12]. we could have stated $\mathbf{a}^*$ as a simple abbreviation of the expression $\lambda P((\check{}P)_{\langle e,t\rangle}(\mathbf{a}_e))$, for all ME_e, without the abstract-operation in (157).

Original rule	Corresponding Montagovian rule
S → NP VP	(P4)
S → SAdv S	(P9)
S → SCP S	(P7)
S → S Conj-S	(P16)
Conj-S → Conj S	(P12)
NP → Det N	(P2)
NP → NP Conj-NP	(P18)
Conj-NP → Conj NP	(P14)
VP → IV	trivial
VP → IV Adv	(P11)
VP → TV NP	(P5)
VP → ICP S	(P8)
VP → Neg VP	(P24)
VP → VP Conj-VP	(P17)
Conj-VP → Conj VP	(P13)
PP → P NP	(P6)
N → AdjP N	(P10)
AdjP → Neg AdjP	(P25)
AdjP → Adj	trivial
AdjP → AdjP Conj-AdjP	(P19)
Conj-AdjP → Conj AdjP	(P15)

Table 3: Transition from CFG to Montagovian Syntax

The type assignment of the expression in (156), by using the subscript notation, is $\left[\lambda P_{\langle s,\langle e,t\rangle\rangle}\left(\left(\check{}P_{\langle s,\langle e,t\rangle\rangle}\right)_{\langle e,t\rangle}(\mathbf{a}_e)\right)_t\right] : \langle\langle s,\langle e,t\rangle\rangle, t\rangle$.

Notation 19. If for some collection of HL expressions $\{\alpha_i\}_{n\geq i\geq 1}$ (with fixed $n \in \mathbb{N}$) we have that $\alpha_1 \xrightarrow{\mathsf{render}} \alpha'_1,\ \ldots,\ \alpha_n \xrightarrow{\mathsf{render}} \alpha'_n$, we abbreviate this as in (158):

$$\alpha_1, \ldots, \alpha_n \xrightarrow{\mathsf{render}} \alpha'_1, \ldots, \alpha'_n \tag{158}$$

Rendering Rules of AGr

(R1) [See (109)–(122) for $\mathrm{B_C}$] If C is a category not in {Det, T, SAdv, Conj, Neg} and $\alpha \in \mathrm{B_C}$, then $\alpha \xrightarrow{\mathsf{render}} \mathbf{c}$, for some $\mathbf{c} \in \mathsf{Const}_{\mathsf{Type(C)}}$

(R2) necessarily $\xrightarrow{\mathsf{render}} (\lambda p\,(\Box\check{}p))$, where $p : \langle s, t\rangle$

(R3) and $\xrightarrow{\mathsf{render}} \wedge$, or $\xrightarrow{\mathsf{render}} \vee$

(R4) does not $\xrightarrow{\mathsf{render}} \neg$

(R5) [See (109) for $\mathrm{B_C}$] If $\alpha \in \mathrm{B_T} - \{\, \mathrm{he}_0, \mathrm{she}_0, \ldots, \mathrm{he}_n, \mathrm{she}_n, \ldots \}$, then:

$$\alpha \xrightarrow{\mathsf{render}} \mathbf{a}^* \tag{159}$$

for some $\mathbf{a} \in \mathsf{Const}_e$

(R6) $\mathrm{he}_n \xrightarrow{\mathsf{render}} (\lambda P\,(P\{x_n\}))$, where $P : \langle s, \langle e, t\rangle\rangle$

(R7) [See (P2)] If $\delta \in \mathrm{P_N}$ and $\delta \xrightarrow{\mathsf{render}} \delta'$, then:

$$F_0(\delta) = \text{every } \delta \xrightarrow{\mathsf{render}} (\lambda P\,(\forall x\,(\delta'(x) \to P\{x\}))) \tag{160}$$

$$F_1(\delta) = \text{the } \delta \xrightarrow{\mathsf{render}} (\lambda P\,(\exists y\,(\forall x\,(\delta'(x) \leftrightarrow x = y) \wedge P\{y\}))) \tag{161}$$

$$F_2(\delta) = \text{a/an } \delta \xrightarrow{\mathsf{render}} (\lambda P\,(\exists x\,(\delta'(x) \wedge P\{x\}))) \tag{162}$$

where $P : \langle s, \langle e, t\rangle\rangle$

(R8) [See (P3)] If $\delta \in \mathrm{P_N}$, $\alpha \in \mathrm{P_S}$ and $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_{3,n}(\delta, \alpha) \xrightarrow{\mathsf{render}} \lambda x_{\,n}(\delta'(x_n) \wedge \alpha')) \tag{163}$$

where $x_n : e$

(R9) [See (P4)] If $\delta \in \mathrm{P_T}$, $\alpha \in \mathrm{P_{IV}}$, with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_5(\delta, \alpha) \xrightarrow{\mathsf{render}} \delta'(\hat{}\,\alpha') \tag{164}$$

(R10) [See (P5)] If $\delta \in \mathrm{P_{TV}}$, $\alpha \in \mathrm{P_T}$ with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_5(\delta, \alpha) \xrightarrow{\mathsf{render}} \delta'(\hat{}\,\alpha') \tag{165}$$

(R11) [See (P6)] If $\delta \in \mathrm{P_P}$, $\alpha \in \mathrm{P_T}$ with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_5(\delta, \alpha) \xrightarrow{\mathsf{render}} \delta'(\hat{}\,\alpha') \tag{166}$$

(R12) [See (P7)] If $\delta \in \mathrm{P_{SCP}}$, $\alpha \in \mathrm{P_S}$ with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_6(\delta, \alpha) \xrightarrow{\mathsf{render}} \delta'(\hat{}\,\alpha') \tag{167}$$

(R13) [See (P8)] If $\delta \in \mathrm{P_{ICP}}$, $\alpha \in \mathrm{P_{IV}}$ with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_6(\delta, \alpha) \xrightarrow{\mathsf{render}} \delta'(\hat{}\,\alpha')$$

(R14) [See (P9)] If $\delta \in \mathrm{P_{SAdv}}$, $\alpha \in \mathrm{P_S}$ with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_6(\delta, \alpha) \xrightarrow{\mathsf{render}} \delta'(\hat{}\,\alpha') \tag{168}$$

(R15) [See (P10)] If $\delta \in \mathrm{P_{Adj}}$, $\alpha \in \mathrm{P_N}$ with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_7(\delta, \alpha) \xrightarrow{\mathsf{render}} \delta'(\hat{}\alpha') \tag{169}$$

(R16) [See (P11)] If $\delta \in \mathrm{P_{Adv}}$, $\alpha \in \mathrm{P_{IV}}$ with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_7(\delta, \alpha) \xrightarrow{\mathsf{render}} \delta'(\hat{}\alpha') \tag{170}$$

(R17) [See (P12)] If $\delta \in \mathrm{P_{Conj}}$, $\alpha \in \mathrm{P_S}$ with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_8(\delta, \alpha) \xrightarrow{\mathsf{render}} \delta'(\alpha') \tag{171}$$

(R18) [See (P13)] If $\delta \in \mathrm{P_{Conj}}$, $\alpha \in \mathrm{P_{IV}}$, with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_8(\delta, \alpha) \xrightarrow{\mathsf{render}} \left[\lambda Q\, (\mathsf{geach}_2^a(\delta')(\alpha')(Q))\right] \tag{172}$$

where $Q : \langle e, t\rangle$
For the operation geach_2^a, see (132a)–(132b) and (134a).

(R19) [See (P14)] If $\delta \in \mathrm{P_{Conj}}$, $\alpha \in \mathrm{P_T}$, with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_8(\delta, \alpha) \xrightarrow{\mathsf{render}} \left[\lambda R\, (\mathsf{geach}_2^b(\delta')(\alpha')(R))\right] \tag{173}$$

where $R : \langle\langle s, \langle e, t\rangle\rangle, t\rangle$.
For the operation geach_2^b, see (132a)–(132b) and (134b).

(R20) [See (P15)] If $\delta \in \mathrm{P_{Conj}}$, $\alpha \in \mathrm{P_{Adj}}$ with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_8(\delta, \alpha) \xrightarrow{\mathsf{render}} \left[\lambda M\, (\mathsf{geach}_2^b(\mathsf{geach}_2^a(\delta'))(\alpha')(M))\right] \tag{174}$$

where $M : \langle\langle s, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle$
See (132a)–(132b), for geach_2^a: (134a); for geach_2^b: (134b).

(R21) [See (P16)] If $\delta \in \mathrm{P_S}$, $\alpha \in \mathrm{P_{Conj\text{-}S}}$ with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_9(\delta, \alpha) \xrightarrow{\mathsf{render}} \alpha'(\delta') \tag{175}$$

(R22) [See (P17)] If $\delta \in \mathrm{P_{IV}}$, $\alpha \in \mathrm{P_{Conj\text{-}IV}}$ with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_9(\delta, \alpha) \xrightarrow{\mathsf{render}} \alpha'(\delta') \tag{176}$$

(R23) [See (P18)] If $\delta \in \mathrm{P_T}$, $\alpha \in \mathrm{P_{Conj\text{-}T}}$ with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_9(\delta, \alpha) \xrightarrow{\mathsf{render}} \alpha'(\delta') \tag{177}$$

(R24) [See (P19)] If $\delta \in \mathrm{P}_{\mathrm{Adj}}$, $\alpha \in \mathrm{P}_{\text{Conj-Adj}}$, with $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_9(\delta, \alpha) \xrightarrow{\mathsf{render}} \alpha'(\delta') \tag{178}$$

(R25) [See (P20)] If $\delta \in \mathrm{P}_{\mathrm{T}}$, $\alpha \in \mathrm{P}_{\mathrm{S}}$ and $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_{10,n}(\delta, \alpha) \xrightarrow{\mathsf{render}} \delta'(\hat{}\lambda x_n\, \alpha') \tag{179}$$

(R26) [See (P21)] If $\delta \in \mathrm{P}_{\mathrm{T}}$, $\alpha \in \mathrm{P}_{\mathrm{N}}$ and $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_{10,n}(\delta, \alpha) \xrightarrow{\mathsf{render}} \big(\lambda y\, (\delta'(\lambda x_n\, (\alpha'(y))))\big) \tag{180}$$

(R27) [See (P22)] If $\delta \in \mathrm{P}_{\mathrm{T}}$, $\alpha \in \mathrm{P}_{\mathrm{IV}}$ and $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_{10,n}(\delta, \alpha) \xrightarrow{\mathsf{render}} \big(\lambda y\, (\delta'(\lambda x_n\, (\alpha'(y))))\big) \tag{181}$$

(R28) [See (P23)] If $\delta \in \mathrm{P}_{\mathrm{T}}$, $\alpha \in \mathrm{P}_{\mathrm{IV}}$ and $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_{11}(\delta, \alpha) \xrightarrow{\mathsf{render}} \mathbf{P}(\delta'(\hat{}\alpha')) \tag{182}$$

(R29) [See (P23)] If $\delta \in \mathrm{P}_{\mathrm{T}}$, $\alpha \in \mathrm{P}_{\mathrm{IV}}$ and $\delta, \alpha \xrightarrow{\mathsf{render}} \delta', \alpha'$, then:

$$F_{12}(\delta, \alpha) \xrightarrow{\mathsf{render}} \mathbf{F}(\delta'(\hat{}\alpha')) \tag{183}$$

(R30) [See (P24)] If $\delta \in \mathrm{P}_{\mathrm{Neg}}$, $\alpha \in \mathrm{P}_{\mathrm{IV}}$ and $\delta, \alpha \xrightarrow{\mathsf{render}} \neg, \alpha'$, then:

$$F_{13}(\delta, \alpha) \xrightarrow{\mathsf{render}} \mathsf{geach}_1^a(\neg)(\alpha) \tag{184}$$

(R31) [See (P25)] If $\delta \in \mathrm{P}_{\mathrm{Neg}}$, $\alpha \in \mathrm{P}_{\mathrm{Adj}}$ and $\delta, \alpha \xrightarrow{\mathsf{render}} \neg, \alpha'$, then:

$$F_{13}(\delta, \alpha) \xrightarrow{\mathsf{render}} \mathsf{geach}_1^b(\mathsf{geach}_1^a(\neg))(\alpha) \tag{185}$$

(R32) [See (P26)] If $\delta \in \mathrm{P}_{\mathrm{Neg}}$, $\alpha \in \mathrm{P}_{\mathrm{S}}$ and $\delta, \alpha \xrightarrow{\mathsf{render}} \neg, \alpha'$ then:

$$F_{13}(\delta, \alpha) \xrightarrow{\mathsf{render}} \neg\alpha' \tag{186}$$

Explanation of (R1)–(R32)

1. (R1)–(R5) are simple rules for rendering the basic expressions into L_{IL}
2. (R7) are rules concerning the renderings of determiners. Clearly, they are very similar to what we have seen so far

3. (R8) This rule concerns "such-that". A variable x_n of type e is abstracted over which yields an expression of type $\langle e, t\rangle$, which corresponds to an expression of category N

4. (R9)–(R16) Are straightforward and work in the same way. For instance, in (R9) the rendering of a term phrase is applied on the intensional version of the rendering of an intransitive verb. The $\hat{}$ is important for the types to match, recall the problems in Section 4.5.2

5. (R17)–(R20) also work in the same way. What for instance (R18) says is that if we render some conjunction, i.e. "and", into $\wedge$ and some intransitive verb, i.e. "runs", into α' we use the appropriate geach-rule on $\wedge$ to make the types match and then render a formula that can be thought of as $\wedge\alpha'$, i.e. half a conjunction that needs another intrasitive verb P to be completed

   Note that (R20) is special in that it uses two applications of the geach rules after each other. This is to transform the $\wedge$ into something of the rather complex type:

   $$\langle\langle\langle s, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle, \langle\langle s, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle\rangle, \langle\langle s, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle\rangle$$

6. (R21)–(R24) complement the rules (R17)–(R20). They add "missing conjunct" mentioned above.

   This is so, because conjunction constants, e.g., "and" and $\wedge$, take their arguments one-by-one, i.e., we have $\wedge(\alpha_2)(\alpha_1)$ instead of the infix binary convjunction symbol $(\alpha_2 \wedge \alpha_1)$. (R25)–(R27) are used when we want to quantify term phrases of category T, i.e., NP, in traditional notations) into phrases for an indexed "pronoun", such as it_0, i.e. when we want to replace it_0 with the term T phrase. The idea is that it_0 will provide some variable x_0 in the rendering expression. We can then abstract over x_0 by λx_0, to provide its replacement with the rendering of the T expressions

7. (R28) and (R29) simply add **P** and **F** to formula, to restrict their truth conditions to a specific set of points in time

8. (R32)–(R31) are simple negation rules. geach-rules are used when needed to make the types match. Again, the treatment of adjectives is requires a string of two geach-rules. First, we raise the type of $\neg$ to $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$ by geach_1^a. After that, we apply geach_1^b to raise the type of this to $\langle\langle\langle s, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle, \langle\langle s, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle\rangle$

That the (R1)–(R32) rules work together with (P1)–(P26) is of course nothing obvious and needs to be proved rigorously. Especially, it is not at all obvious that the rules (R17)–(R24) together with the corresponding (P12)–(P19) work the way we want them to. Here I prove Proposition 4.4, which covers the base case for the rules for coordination of adjectives. Propositions concerning the other rules and their proofs are similar but are left for future work.

Proposition 4.4. For readability, let $\sigma = \langle\langle s, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle$. Assume (187a)–(187b), according to Table 2:

$$\mathrm{A}_1, \mathrm{A}_2 \in B_{\mathrm{Adj}} \tag{187a}$$

$$\begin{gathered}\mathrm{A}_1, \mathrm{A}_2 \xrightarrow{\mathsf{render}} \alpha_1, \alpha_2 \\ \text{for } \alpha_1, \alpha_2 : \sigma\end{gathered} \tag{187b}$$

Then:

$$\begin{gathered}\mathrm{A}_1 \text{ and } \mathrm{A}_2 \xrightarrow{\mathsf{render}} \\ \Big(\lambda y_{\langle s,\langle e,t\rangle\rangle}\,(\lambda x_e\,((\alpha_1\,(y_{\langle s,\langle e,t\rangle\rangle})(x_e)) \wedge (\alpha_2(y_{\langle s,\langle e,t\rangle\rangle})(x_e))))\Big) \;:\; \sigma\end{gathered} \tag{188}$$

Proof. First, we have that $\mathrm{A}_1, \mathrm{A}_2 \in \mathrm{P}_{\mathrm{Adj}}$ by (P1). Also:

$$\text{and} \in \mathrm{P}_{\mathrm{Conj}} \tag{189}$$

So:

$$\text{and } \mathrm{A}_2 \in \mathrm{P}_{\text{Conj-Adj}} \qquad \text{by (P15)} \tag{190}$$

$$\mathrm{A}_1 \text{ and } \mathrm{A}_2 \in \mathrm{P}_{\mathrm{Adj}} \qquad \text{by (P19)} \tag{191}$$

And so we can apply the corresponding rendering rules. I indicate which rule is applied after each rendering.

$$\mathrm{A}_1, \mathrm{A}_2 \xrightarrow{\mathsf{render}} \alpha_1, \alpha_2 \tag{192a}$$

$$\text{and} \xrightarrow{\mathsf{render}} \wedge \qquad \text{(R3)} \tag{192b}$$

$$\text{and } \mathrm{A}_2 \xrightarrow{\mathsf{render}} \Big[\lambda M_\sigma\,\big(\mathsf{geach}_2^b(\mathsf{geach}_2^a(\wedge))(\alpha_2)(M_\sigma)\big)\Big] \qquad \text{(R20)} \tag{192c}$$

Let $K = \mathsf{geach}_2^a(\wedge)$. We then have, by (134a) and (134b):

$$K = \Big[\lambda P_{\langle e,t\rangle}\,(\lambda Q_{\langle e,t\rangle}\,(\lambda x_e\,(\wedge(P_{\langle e,t\rangle}(x))(Q_{\langle e,t\rangle}(x)))))\Big] \tag{192d}$$

$$\begin{gathered}\mathsf{geach}_2^b(K) = \\ (\lambda R_\sigma\,(\lambda T_\sigma\,(\lambda y_{\langle s,\langle e,t\rangle\rangle}\,(K(R_\sigma(y_{\langle s,\langle e,t\rangle\rangle}))(T_\sigma(y_{\langle s,\langle e,t\rangle\rangle}))))))\end{gathered} \tag{192e}$$

hence

$$\begin{gathered}\mathsf{geach}_2^b(K)(\alpha_2)(M_\sigma) \\ \xrightarrow{\mathsf{convert}} (\lambda y_{\langle s,\langle e,t\rangle\rangle}\,(K(\alpha_2(y_{\langle s,\langle e,t\rangle\rangle}))(M_\sigma(y_{\langle s,\langle e,t\rangle\rangle}))))\end{gathered} \tag{192f}$$

By using convert on (192d):

$$\begin{gathered}K(\alpha_2(y_{\langle s,\langle e,t\rangle\rangle}))(M_\sigma(y_{\langle s,\langle e,t\rangle\rangle})) = \\ \Big[\lambda P_{\langle e,t\rangle}\,(\lambda Q_{\langle e,t\rangle}\,(\lambda x_e\,(\wedge(P_{\langle e,t\rangle}(x_e))(Q_{\langle e,t\rangle}(x_e)))))\Big] \\ \Big(\alpha_2(y_{\langle s,\langle e,t\rangle\rangle})\Big)\Big(M_\sigma(y_{\langle s,\langle e,t\rangle\rangle})\Big) \\ \xrightarrow{\mathsf{convert}} (\lambda x_e\,(\wedge(\alpha_2(y_{\langle s,\langle e,t\rangle\rangle})(x_e))(M_\sigma(y_{\langle s,\langle e,t\rangle\rangle})(x_e))))\end{gathered} \tag{192g}$$

And so, the rendering expression in (192c) is converted to:

$$\begin{aligned}&\Big[\lambda M_\sigma\big(\mathsf{geach}_2^b(\mathsf{geach}_2^b(\wedge))\big)(\alpha_2)(M_\sigma)\Big]\\&\qquad\xrightarrow{\text{convert}}\Big[\lambda M_\sigma\Big(\lambda y_{\langle s,\langle e,t\rangle\rangle}\,(\lambda x_e\,(\wedge(\alpha_2(y_{\langle s,\langle e,t\rangle\rangle})(x_e))\\&\qquad\qquad(M_\sigma(y_{\langle s,\langle e,t\rangle\rangle})(x_e))))\Big)\Big]\end{aligned}\tag{192h}$$

And so finally:

$$\begin{aligned}&\text{A}_1\text{ and A}_2\\&\qquad\xrightarrow{\text{render}}\Big[\lambda M_\sigma\Big(\lambda y_{\langle s,\langle e,t\rangle\rangle}\,(\lambda x_e\,(\wedge(\alpha_2(y_{\langle s,\langle e,t\rangle\rangle})(x_e))\quad\text{(R24)}\\&\qquad\qquad(M_\sigma\,(y_{\langle s,\langle e,t\rangle\rangle})(x_e))))\Big)\Big](\alpha_1)\end{aligned}\tag{192i}$$

$$\begin{aligned}&\xrightarrow{\text{convert}}\Big(\lambda y_{\langle s,\langle e,t\rangle\rangle}\,(\lambda x_e\,(\wedge(\alpha_2(y_{\langle s,\langle e,t\rangle\rangle})(x_e))\\&\qquad\qquad(\alpha_1\,(y_{\langle s,\langle e,t\rangle\rangle})(x_e))))\Big)=\\&\Big(\lambda y_{\langle s,\langle e,t\rangle\rangle}\,(\lambda x_e\,((\alpha_1\,(y_{\langle s,\langle e,t\rangle\rangle})(x_e))\wedge(\alpha_2(y_{\langle s,\langle e,t\rangle\rangle})(x_e))))\Big)\;:\;\sigma\end{aligned}\tag{192j}$$

□

The best way of understanding these rules, however, is to see them in action. In the following section, I shall give examples of renderings using (R1)–(R32). By them, I will show how the previously mentioned limitations of L_1 have been overcome, at least to some desirable extend.

4.6.3 Restricting the Models of L_{IL}

One important concept in Montauge semantics is that of *meaning postulates*. Essentially, a meaning postulate is a formula that is required to hold in each legitimate model. Effectively, it imposes some semantic property for the expressions involved by restricting our choice of models to those in which this property holds. Montague [12] introduced several meaning postulates in very general forms, by using variables of certain types. For clarity, instead of for generality, I shall only give specific instances of meaning postulates that I rely on in the coming sections.

Recall the brace notation in (79) and (80).

MP1 $\forall x\,\forall P\,\Box\,(\mathbf{greater\text{-}than}(P)(x)\leftrightarrow[\check{}P]\,(\hat{}\lambda y\,([\hat{}\mathbf{greater\text{-}than}_*]\{x,y\})))$

MP2 $\forall x\,\forall P\,\Box\,(\mathbf{write}(P)(x)\leftrightarrow[\check{}P]\,(\hat{}\lambda y\,([\hat{}\mathbf{write}_*]\{x,y\})))$

where:

$$x,y:e\tag{193a}$$

$$P : \langle s, \langle\langle s, \langle e, t\rangle\rangle, t\rangle\rangle, \tag{193b}$$

$$p : \langle s, t\rangle \tag{193c}$$

$$\textbf{greater-than}_*, \textbf{write}_* \in \mathsf{Const}_{\langle e,\langle e,t\rangle\rangle} \tag{193d}$$

$$\tag{193e}$$

The new constants labelled with "$_*$" are the extensional versions of the original constants to which they correspond.

4.7 Rendering English into L_{IL}

We are now ready to deal with the problems we encountered in Section 3.6. We shall show how we can use the syntactic analyses and rendering of various expressions of English into L_{IL}, for semantic representations.

We can often use metavariables, such as P, Q, R with typical type assignments (194a)–(194d), unless their types are specified in other ways:

$$x_n, x, y, z \in \mathsf{Vars}_e \quad \text{for entities} \tag{194a}$$

$$p \in \mathsf{Vars}_{\langle s,t\rangle} \quad \text{for propositions} \tag{194b}$$

$$P, Q \in \mathsf{Vars}_{\langle s,\langle e,t\rangle\rangle} \quad \text{for properties of entities} \tag{194c}$$

$$R \in \mathsf{Vars}_{\langle e,t\rangle} \quad \text{for sets of entities} \tag{194d}$$

In the coming calculations, I will indicate which of the rules (R1)–(R31) I rely on, when I use the $\xrightarrow{\mathsf{render}}$ operation. After each rendering, I will also indicate the type of the rendered expression. I will let $\xrightarrow{\mathsf{MP1}}$ and $\xrightarrow{\mathsf{MP2}}$ denote applications of MP1 and MP2, respectively. Sometimes, I will indicate the types of expressions by adding subscripts to them. However, I shall only do this where I consider it to increase readability. I will also use brackets of different kinds and sizes — also this is for readablity.

Generally, the calculations of the rendering of a compound expression A from the renderings of its components are non-deterministic, but always leading to the same end result. I.e., in many cases there are different choices of the order of applications of render rules on components, λ-conversion, λ-abstraction, and up-down cancellation. Such reductions are applyed until no more reductions are possible. All alterantive reductions end with expressions of L_{IL}, which are equivalent with respect to renaming bound varaiables. We usually choose reduction steps that simplify the sub-components, as soon as possible for transperency. One may choose some other systematic strategies, but they may laed to more complex intermediate expressions. E.g., we could λ-abstract after each rendering of a component of an expression A that is of the syntactice category S for a sentence. In that way, we usually end up with a rather complex expression when we reach the top expression of category S, which we would then simplify using a string of λ-conversions and down-up cancellations.

4.7.1 Predication

Let us once again consider:

S4 Jacques sings well

Recall, the problem was that there was no way in L_1 for "well" to modify "sing". We have solved this problem here, as seen by the following rendering:

$$\text{Jacques} \xrightarrow{\text{render}} \mathbf{j}^* : \langle\langle s, \langle e, t\rangle\rangle, t\rangle \quad \text{(R5)} \quad \text{(195a)}$$

$$\xrightarrow{\text{abstract}} \left[\lambda P\left(\left(\check{}P\right)_{\langle e,t\rangle}(\mathbf{j}_e)\right)\right] \quad \text{(195b)}$$

$$\text{sings} \xrightarrow{\text{render}} \mathbf{sing} : \langle e, t\rangle \quad \text{(R1)} \quad \text{(195c)}$$

$$\text{well} \xrightarrow{\text{render}} \mathbf{well} : \langle\langle s, \langle e, t\rangle\rangle, \langle e, t\rangle\rangle \quad \text{(R1)} \quad \text{(195d)}$$

$$\text{sing well} \xrightarrow{\text{render}} \mathbf{well}(\hat{}\mathbf{sing}) : \langle e, t\rangle \quad \text{(R16)} \quad \text{(195e)}$$

$$\text{Jacques sings well} \xrightarrow{\text{render}} \Big[\lambda P\left(\check{}P(\mathbf{j})\right)\Big](\hat{}\mathbf{well}(\hat{}\mathbf{sing}))_{\langle s,\langle e,t\rangle\rangle} : t \quad \text{(R9)} \quad \text{(195f)}$$

$$\Big[\lambda P\left(\check{}P(\mathbf{j})\right)\Big](\hat{}\mathbf{well}(\hat{}\mathbf{sing})) \xrightarrow{\text{convert}} \left(\check{}(\hat{}\mathbf{well}(\hat{}\mathbf{sing}))\right)(\mathbf{j}) \quad \text{(195g)}$$

$$\xrightarrow{\text{cancel}} \mathbf{well}(\hat{}\mathbf{sing})(\mathbf{j}) \quad \text{(195h)}$$

And so we have a satisfactory rendering of the sentence.

4.7.2 Quantification

Here we return to:

S5 Every Belgian sings

We get:

$$\text{Belgian} \xrightarrow{\text{render}} \mathbf{belgian} : \langle e, t\rangle \quad \text{(R1)} \quad \text{(196a)}$$

$$\text{Every Belgian} \xrightarrow{\text{render}} \Big[\lambda P\left(\forall x\left(\mathbf{belgian}(x) \to \check{}P(x)\right)\right)\Big] : \langle\langle s, \langle e, t\rangle\rangle, t\rangle \quad \text{(R7)} \quad \text{(196b)}$$

$$\text{sing} \xrightarrow{\text{render}} \mathbf{sing} : \langle e, t\rangle \quad \text{(R1)} \quad \text{(196c)}$$

$$\text{Every Belgian sings} \xrightarrow{\text{render}} \Big[\lambda P\left(\forall x\left(\mathbf{belgian}(x) \to \check{}P(x)\right)\right)\Big](\hat{}\mathbf{sing})_{\langle s,\langle e,t\rangle\rangle} : t \quad \text{(R9)} \quad \text{(196d)}$$

$$\Big[\lambda P\left(\forall x\left(\mathbf{belgian}(x) \to \check{}P(x)\right)\right)\Big](\hat{}\mathbf{sing})_{\langle s,\langle e,t\rangle\rangle} \xrightarrow{\text{convert}} \forall x\left(\mathbf{belgian}(x) \to \check{}(\hat{}\mathbf{sing}(x))\right) \quad \text{(196e)}$$

$$\xrightarrow{\text{cancel}} \forall x\left(\mathbf{belgian}(x) \to \mathbf{sing}(x)\right) \quad \text{(196f)}$$

Recall, in L_1 there were problems with rendering quantified NPs in a compositional way from their components. There were also problems with putting combinng them with the renderings of VPs. This problem has been solved here. Furthermore, we see that AGr treats NPs which are proper nouns and quantifier NPs uniformly. In both cases, the NP is rendered into an L_{IL}-expression of type $\langle\langle s, \langle e, t\rangle\rangle, t\rangle$, whereas the expressions simplify to forms that are similar or the same as those in FOL.

Let us now consider the trickier sentence S2:

S2 Every integer is greater than some rational number

Montauge provides us with a way to account for both readings. Here we show how to get the de dicto reading:

rational number $\xrightarrow{\text{render}}$ **rational-number** : $\langle e, t\rangle$ (R1) (197a)

some rational number
$\xrightarrow{\text{render}} \left[\lambda P\left(\exists y\left(\textbf{rational-number}(y) \wedge {}^{\vee}P(y)\right)\right)\right]$: (R7) (197b)
$\langle\langle s, \langle e, t\rangle\rangle, t\rangle$

is greater than
$\xrightarrow{\text{render}}$ **greater-than** : $\langle\langle s, \langle\langle s, \langle e, t\rangle\rangle, t\rangle\rangle, \langle e, t\rangle\rangle$ (R1) (197c)

is greater than some rational number
$\xrightarrow{\text{render}}$ **greater-than**
$\left({}^{\wedge}\left[\lambda P\left(\exists y\left(\textbf{rational-number}(y) \wedge {}^{\vee}P(y)\right)\right)\right]\right) : \langle e, t\rangle$ (R10) (197d)

integer $\xrightarrow{\text{render}}$ **integer** : $\langle e, t\rangle$ (R1) (197e)

Every integer
$\xrightarrow{\text{render}} \left[\lambda Q\left(\forall x\left(\textbf{integer}(x) \rightarrow {}^{\vee}Q(x)\right)\right)\right] : \langle\langle s, \langle e, t\rangle\rangle, t\rangle$ (R7) (197f)

Every integer is greater than some rational number
$\xrightarrow{\text{render}} \left[\lambda Q\left(\forall x\left(\textbf{integer}(x) \rightarrow {}^{\vee}Q(x)\right)\right)\right]$
$\left({}^{\wedge}\textbf{greater-than}\right.$
$\left.\left({}^{\wedge}\left[\lambda P\left(\exists y\left(\textbf{rational-number}(y) \wedge {}^{\vee}P(y)\right)\right)\right]\right)\right) : t$ (R9) (197g)

$$\xrightarrow{\text{convert}} \forall x\,(\mathbf{integer}(x) \rightarrow \check{}\Big(\hat{}\mathbf{greater\text{-}than}\,(\hat{}\big[\lambda P\,(\exists y\,(\mathbf{rational\text{-}number}(y) \wedge \check{}P(y)))\big]\big)\Big)(x)) \tag{197h}$$

$$\xrightarrow{\text{cancel}} \forall x\,(\mathbf{integer}(x) \rightarrow \mathbf{greater\text{-}than}\,(\hat{}\big[\lambda P\,(\exists y\,(\mathbf{rational\text{-}number}(y) \wedge \check{}P(y)))\big]\big)(x)) \tag{197i}$$

$$\xrightarrow{\text{MP1}} \forall x\,(\mathbf{integer}(x) \rightarrow \check{}\Big(\hat{}\big[\ \lambda P\,(\exists y\,(\mathbf{rational\text{-}number}(y) \wedge \check{}P(y)))\big]\Big)\Big(\hat{}\big[\lambda z\,(\hat{}\mathbf{greater\text{-}than}_*\{x, z\})\big]\Big)) \tag{197j}$$

$$\xrightarrow{\text{cancel}} \forall x\,(\mathbf{integer}(x) \rightarrow \Big(\lambda P\,(\exists y\,(\mathbf{rational\text{-}number}(y) \wedge \check{}P(y)))\Big)\Big(\hat{}\big[\lambda z\,(\hat{}\mathbf{greater\text{-}than}_*\{x, z\})\big]\Big)) \tag{197k}$$

$$\xrightarrow{\text{convert}} \forall x\,(\mathbf{integer}(x) \rightarrow \Big(\exists y\,(\mathbf{rational\text{-}number}(y) \wedge \check{}\Big(\hat{}\big[\lambda z\,(\hat{}\mathbf{greater\text{-}than}_*\{x, z\})\big]\Big)(y))\Big)) \tag{197l}$$

$$\xrightarrow{\text{cancel}} \forall x\,(\mathbf{integer}(x) \rightarrow \Big(\exists y\,(\mathbf{rational\text{-}number}(y) \wedge \big[\lambda z\,(\hat{}\mathbf{greater\text{-}than}_*\{x, z\})\big](y))\Big)) \tag{197m}$$

$$\xrightarrow{\text{convert}} \forall x\,(\mathbf{integer}(x) \rightarrow \exists y\,(\mathbf{rational\text{-}number}(y) \wedge \hat{}\mathbf{greater\text{-}than}_*\{x, y\})) \tag{197n}$$

$$\xrightarrow{\text{cancel}} \forall x\,(\mathbf{integer}(x) \rightarrow \exists y\,(\mathbf{rational\text{-}number}(y) \wedge \mathbf{greater\text{-}than}_*(y)(x))) \tag{197o}$$

We can also get the de re reading easily. Recall that this is considered to be the sentence S2*:

S2* Some rational number, every integer is greater than it_0

We get:

$$\text{it}_0 \xrightarrow{\text{render}} \left[\lambda P\,(\check{}P(x_0))\right] : \langle\langle s, \langle e, t\rangle\rangle, t\rangle \tag{R6} \quad (198a)$$

is greater than

$$\xrightarrow{\text{render}} \mathbf{greater\text{-}than} : \langle\langle s, \langle\langle s, \langle e, t\rangle\rangle, t\rangle\rangle, \langle e, t\rangle\rangle \tag{R1} \quad (198b)$$

is greater than it_0

$$\xrightarrow{\text{render}} \mathbf{greater\text{-}than}\big(\hat{}\left[\lambda P\,(\check{}P(x_0))\right]\big) : \langle e, t\rangle \tag{R10} \quad (198c)$$

$$\text{integer} \xrightarrow{\text{render}} \mathbf{integer} : \langle e, t\rangle \tag{R1} \quad (198d)$$

Every integer

$$\xrightarrow{\text{render}} \Big[\lambda Q\,(\forall x\,(\mathbf{integer}(x) \rightarrow \check{}Q(x)))\Big] : \langle\langle s, \langle e, t\rangle\rangle, t\rangle \tag{R7} \quad (198e)$$

Every integer is greater than it_0

$$\xrightarrow{\text{render}} \Big[\lambda Q\,(\forall x\,(\mathbf{integer}(x) \rightarrow \check{}Q(x)))\Big] \Big(\hat{}\mathbf{greater\text{-}than}\big(\hat{}\left[\lambda P\,(\check{}P(x_0))\right]\big)\Big) : t \tag{R9} \quad (198f)$$

$$\xrightarrow{\text{convert}} \forall x\,(\mathbf{integer}(x) \rightarrow \check{}\Big(\hat{}\mathbf{greater\text{-}than}\big(\hat{}\left[\lambda P\,(\check{}P(x_0))\right]\big)\Big)(x)) \tag{198g}$$

$$\xrightarrow{\text{cancel}} \forall x\,(\mathbf{integer}(x) \rightarrow \mathbf{greater\text{-}than}\big(\hat{}\left[\lambda P\,(\check{}P(x_0))\right]\big)(x)) \tag{198h}$$

$$\xrightarrow{\text{MP1}} \forall x\,(\mathbf{integer}(x) \rightarrow \check{}\big(\hat{}\left[\lambda P\,(\check{}P(x_0))\right]\big)\big(\hat{}\left[\lambda z\,(\hat{}\mathbf{greater\text{-}than}_*\{x, z\})\right]\big)) \tag{198i}$$

$$\xrightarrow{\text{cancel}} \forall x\,(\mathbf{integer}(x) \rightarrow \left[\lambda P\,(\check{}P(x_0))\right]\big(\hat{}\left[\lambda z\,(\hat{}\mathbf{greater\text{-}than}_*\{x, z\})\right]\big)) \tag{198j}$$

$$\xrightarrow{\text{convert}} \forall x\,(\mathbf{integer}(x) \rightarrow \check{}\big(\hat{}\big[\lambda z\,(\hat{}\mathbf{greater\text{-}than}_*\{x,z\})\big]\big)(x_0)) \tag{198k}$$

$$\xrightarrow{\text{cancel}} \forall x\,(\mathbf{integer}(x) \rightarrow \big[\lambda z\,(\hat{}\mathbf{greater\text{-}than}_*\{x,z\})\big](x_0)) \tag{198l}$$

$$\xrightarrow{\text{convert}} \forall x\,(\mathbf{integer}(x) \rightarrow \hat{}\mathbf{greater\text{-}than}_*\{x,x_0\}) \tag{198m}$$

$$\xrightarrow{\text{cancel}} \forall x\,(\mathbf{integer}(x) \rightarrow \mathbf{greater\text{-}than}_*(x_0)(x)) \tag{198n}$$

rational number $\xrightarrow{\text{render}}$ $\mathbf{rational\text{-}number} : \langle e,t\rangle$ (R1) (198o)

some rational number

$$\xrightarrow{\text{render}} \Big[\lambda P\,\big(\exists y(\mathbf{rational\text{-}number}(y) \wedge \check{}P(y))\big)\Big] : \langle\langle s,\langle e,t\rangle\rangle,t\rangle \tag{R7} \quad (198p)$$

Every integer is greater than some rational number

$$\xrightarrow{\text{render}} \Big[\lambda P\,\big(\exists y(\mathbf{rational\text{-}number}(y) \wedge \check{}P(y))\big)\Big]\Big(\hat{}\big[\lambda x_0\,(\forall x\,(\mathbf{integer}(x) \rightarrow \mathbf{greater\text{-}than}_*(x_0)(x)))\big]\Big) : t \tag{R25} \quad (198q)$$

$$\xrightarrow{\text{convert}} \exists y(\mathbf{rational\text{-}number}(y) \wedge \check{}\Big(\hat{}\big[\lambda x_0\,(\forall x\,(\mathbf{integer}(x) \rightarrow \mathbf{greater\text{-}than}_*(x_0)(x)))\big]\Big)(y)) \tag{198r}$$

$$\xrightarrow{\text{cancel}} \exists y(\mathbf{rational\text{-}number}(y) \wedge \big[\lambda x_0\,(\forall x\,(\mathbf{integer}(x) \rightarrow \mathbf{greater\text{-}than}_*(x_0)(x)))\big](y)) \tag{198s}$$

$$\xrightarrow{\text{convert}} \exists y(\mathbf{rational\text{-}number}(y) \wedge \forall x\,(\mathbf{integer}(x) \rightarrow \mathbf{greater\text{-}than}_*(y)(x))) \tag{198t}$$

4.7.3 Tense

We return to S3. Recall the discussion concerning the treatment of tense in L_1. This problem disappears if we render into L_{IL}:

$$\text{Jacques} \xrightarrow{\text{render}} \mathbf{j}^* : \langle\langle s, \langle e, t\rangle\rangle, t\rangle \qquad \text{(R5)} \qquad (199a)$$

$$\xrightarrow{\text{abstract}} \left[\lambda P\,(\check{}P(\mathbf{j}))\right] \qquad (199b)$$

$$\text{sing} \xrightarrow{\text{render}} \mathbf{sing} : \langle e, t\rangle \qquad \text{(R1)} \qquad (199c)$$

$$\text{Jacques sang} \xrightarrow{\text{render}} \mathbf{P}\big(\left[\lambda P\,(\check{}P(\mathbf{j}))\right](\hat{}\mathbf{sing})\big) : t \qquad \text{(R28)} \qquad (199d)$$

$$\mathbf{P}\big(\left[\lambda P\,(\check{}P(\mathbf{j}))\right](\hat{}\mathbf{sing})\big) \xrightarrow{\text{convert}} \mathbf{P}(\check{}(\hat{}\mathbf{sing})(\mathbf{j})) \qquad (199e)$$

$$\xrightarrow{\text{cancel}} \mathbf{P}(\mathbf{sing}(\mathbf{j})) \qquad (199f)$$

So by (199a)–(199e) we have:

$$\text{Jacques sang} \xrightarrow{\text{render}} \mathbf{P}(\mathbf{sing}(\mathbf{j})) \qquad (200)$$

which is precisely what we want. Of course, by an almost identical derivation we get:

$$\text{Jacques will sing} \xrightarrow{\text{render}} \mathbf{F}(\mathbf{sing}(\mathbf{j})) \qquad (201)$$

4.7.4 Modality

Let us now try to deal with:

S12 Necessarily, Jacques sings

We get:

$$\text{Jacques} \xrightarrow{\text{render}} \mathbf{j}^* : \langle\langle s, \langle e, t\rangle\rangle, t\rangle \qquad \text{(R5)} \qquad (202a)$$

$$\xrightarrow{\text{abstract}} \left[\lambda P\,(\check{}P(\mathbf{j}))\right] \qquad (202b)$$

$$\text{sing} \xrightarrow{\text{render}} \mathbf{sing} : \langle e, t\rangle \qquad \text{(R1)} \qquad (202c)$$

$$\text{Jacques sings} \xrightarrow{\text{render}} \left[\lambda P\,(\check{}P(\mathbf{j}))\right](\hat{}\mathbf{sing}) : t \qquad \text{(R9)} \qquad (202d)$$

$$\left[\lambda P\,(\check{}P(\mathbf{j}))\right](\hat{}\mathbf{sing}) \xrightarrow{\text{convert}} \check{}(\hat{}\mathbf{sing})(\mathbf{j}) \xrightarrow{\text{cancel}} \mathbf{sing}(\mathbf{j}) \qquad (202e)$$

$$\text{Necessarily} \xrightarrow{\text{render}} \left[\lambda p\,(\Box\,\check{}p)\right] : \langle\langle s, t\rangle, t\rangle \qquad \text{(R2)} \qquad (202f)$$

$$\text{Necessarily, Jacques sings} \xrightarrow{\text{render}} \left[\lambda p\,(\Box\,\check{}p)\right](\hat{}\mathbf{sing}(\mathbf{j})) : t \qquad \text{(R14)} \qquad (202g)$$

$$\left[\lambda p\,(\Box\,\check{}p)\right](\hat{}\mathbf{sing}(\mathbf{j})) \xrightarrow{\text{convert}} \Box\big(\check{}(\hat{}\mathbf{sing}(\mathbf{j}))\big) \xrightarrow{\text{cancel}} \Box\,(\mathbf{sing}(\mathbf{j})) \qquad (202h)$$

So by (202a)–(202h) we have:

$$\text{Necessarily, Jacques sings} \xrightarrow{\text{render}} \Box\,(\mathbf{sing}(\mathbf{j})) \qquad (203)$$

which is what we wanted.

4.7.5 Intensionality

We return to the sentences S10 and S11. Here we make life easy for us and consider "silly books" and "the author of Being and Time" as proper names and use the present tense "write" rather than "wrote". I.e., we rewrite the sentences as in S10* and S11*

S10* Ludwig thinks that Martin Heidegger writes silly-books

S11* Ludwig thinks that the author of Being and Time writes silly-books

We start by rendering S10*:

$$\text{writes} \xrightarrow{\textsf{render}} \mathbf{write} : \langle\langle s, \langle\langle s, \langle e, t\rangle\rangle, t\rangle\rangle, \langle e, t\rangle\rangle \quad \text{(R1)} \quad \text{(204a)}$$

$$\text{silly-books} \xrightarrow{\textsf{render}} \mathbf{b}^* : \langle\langle s, \langle e, t\rangle\rangle, t\rangle \quad \text{(R5)} \quad \text{(204b)}$$

$$\xrightarrow{\textsf{abstract}} \left[\lambda P\left(\check{}P\left(\mathbf{b}\right)\right)\right] \quad \text{(204c)}$$

$$\text{writes silly-books} \xrightarrow{\textsf{render}} \mathbf{write}\left(\hat{}\left[\lambda P\left(\check{}P\left(\mathbf{b}\right)\right)\right]\right) : \langle e, t\rangle \quad \text{(R10)} \quad \text{(204d)}$$

$$\text{Martin Heidegger} \xrightarrow{\textsf{render}} \mathbf{m}^* : \langle\langle s, \langle e, t\rangle\rangle, t\rangle \quad \text{(R5)} \quad \text{(204e)}$$

$$\xrightarrow{\textsf{abstract}} \left[\lambda Q\left(\check{}Q\left(\mathbf{m}\right)\right)\right] \quad \text{(204f)}$$

$$\text{Martin Heidegger writes silly-books}$$
$$\xrightarrow{\textsf{render}} \left[\lambda Q\left(\check{}Q(\mathbf{m})\right)\right]\left(\hat{}\mathbf{write}\left(\hat{}\left[\lambda P\left(\check{}P\left(\mathbf{b}\right)\right)\right]\right)\right) : t \quad \text{(R9)} \quad \text{(204g)}$$

$$\xrightarrow{\textsf{convert}} \check{}\left(\hat{}\mathbf{write}\left(\hat{}\left[\lambda P\left(\check{}P\left(\mathbf{b}\right)\right)\right]\right)\right)(\mathbf{m}) \quad \text{(204h)}$$

$$\xrightarrow{\textsf{cancel}} \mathbf{write}\left(\hat{}\left[\lambda P\left(\check{}P\left(\mathbf{b}\right)\right)\right]\right)(\mathbf{m}) \quad \text{(204i)}$$

$$\xrightarrow{\textsf{MP2}} \check{}\left(\hat{}\left[\lambda P\left(\check{}P\left(\mathbf{b}\right)\right)\right]\right)\left(\hat{}\left[\lambda z(\hat{}\mathbf{write}_*\{\mathbf{m}, z\})\right]\right) \quad \text{(204j)}$$

$$\xrightarrow{\textsf{cancel}} \left[\lambda P\left(\check{}P\left(\mathbf{b}\right)\right)\right]\left(\hat{}\left[\lambda z(\hat{}\mathbf{write}_*\{\mathbf{m}, z\})\right]\right) \quad \text{(204k)}$$

$$\xrightarrow{\textsf{convert}} \check{}\left(\hat{}\left[\lambda z\left(\hat{}\mathbf{write}_*\{\mathbf{m}, z\}\right)\right]\right)(\mathbf{b}) \quad \text{(204l)}$$

$$\xrightarrow{\textsf{cancel}} \lambda z\left(\hat{}\mathbf{write}_*\{\mathbf{m}, z\}\right)(\mathbf{b}) \quad \text{(204m)}$$

$$\xrightarrow{\textsf{cancel}} \lambda z\left(\mathbf{write}_*(\mathbf{m}, z)\right)(\mathbf{b}) \quad \text{(204n)}$$

$$\xrightarrow{\textsf{convert}} \mathbf{write}_*(\mathbf{m}, \mathbf{b}) \quad \text{(204o)}$$

$$\text{thinks that} \xrightarrow{\textsf{render}} \mathbf{think\text{-}that} : \langle\langle s, t\rangle, \langle e, t\rangle\rangle \quad \text{(R1)} \quad \text{(204p)}$$

$$\text{thinks that Martin Heidegger writes silly-books} \xrightarrow{\text{render}} \mathbf{think\text{-}that}(\hat{}\mathbf{write}_*(\mathbf{m},\mathbf{b})) : \langle e,t\rangle \quad \text{(R12)} \quad \text{(204q)}$$

$$\text{Ludwig} \xrightarrow{\text{render}} \mathbf{l}^* : \langle\langle s,\langle e,t\rangle\rangle,t\rangle \quad \text{(R5)} \quad \text{(204r)}$$

$$\xrightarrow{\text{abstract}} \left[\lambda Q\,(\check{}Q\,(\mathbf{l}))\right] \quad \text{(204s)}$$

$$\text{Luwdig thinks that Martin Heidegger writes silly-books} \xrightarrow{\text{render}} \left[\lambda Q\,(\check{}Q\,(\mathbf{l}))\right](\hat{}\mathbf{think\text{-}that}(\hat{}\mathbf{write}_*(\mathbf{m},\mathbf{b}))) : t \quad \text{(R9)} \quad \text{(204t)}$$

$$\xrightarrow{\text{convert}} \check{}(\hat{}\mathbf{think\text{-}that}(\hat{}\mathbf{write}_*(\mathbf{m},\mathbf{b})))(\mathbf{l}) \quad \text{(204u)}$$

$$\xrightarrow{\text{cancel}} \mathbf{think\text{-}that}(\hat{}\mathbf{write}_*(\mathbf{m},\mathbf{b}))(\mathbf{l}) \quad \text{(204v)}$$

And so we have:

$$\text{Ludwig thinks that Martin Heidegger writes silly-books} \xrightarrow{\text{render}} \mathbf{think\text{-}that}(\hat{}\mathbf{write}_*(\mathbf{m},\mathbf{b}))(\mathbf{l}) \quad (205)$$

In a similar manner, if we let:

$$\text{The author of Being and Time} \xrightarrow{\text{render}} \mathbf{a}^* \quad (206)$$

and proceed in the same way as in (204a)–(204v) we get:

$$\text{Ludwig thinks that the author of time writes silly-books} \xrightarrow{\text{render}} \mathbf{think\text{-}that}(\hat{}\mathbf{write}_*(\mathbf{a},\mathbf{b}))(\mathbf{l}) \quad (207)$$

Now, even if $[\![\mathbf{m}]\!]^{\mathcal{M},w,t,h} = [\![\mathbf{a}]\!]^{\mathcal{M},w,t,h}$ for some $(w,t) \in W \times T$, a substitution is blocked by the intensional$\hat{}$. In other words, the restricted version of substitution — see Proposition 4.2 — does not apply here. $[\![\hat{}\mathbf{write}_*(\mathbf{m},\mathbf{b})]\!]^{\mathcal{M},w,t,h}$ and $[\![\hat{}\mathbf{write}_*(\mathbf{a},\mathbf{b})]\!]^{\mathcal{M},w,t,h}$ denote two entirely different functions. I.e., as a counterexample, we can construct a model with some $(x,y) \in W \times T$, such that:

$$[\![\mathbf{b}]\!]^{\mathcal{M},x,y,h} = I(\mathbf{b})(x,y) \neq I(\mathbf{a})(x,y) = [\![\mathbf{a}]\!]^{\mathcal{M},x,y,h} \quad (208a)$$

$$[\![\mathbf{write}_*(\mathbf{m},\mathbf{b})]\!]^{\mathcal{M},x,y,h} \neq [\![\mathbf{write}_*(\mathbf{m},\mathbf{a})]\!]^{\mathcal{M},x,y,h} \quad (208b)$$

5 Summary and Outlook

In this thesis I have introduced Montague's L_{IL} and provided a formal syntax in a grammar AGr which, for a small fragment of English, represents key phrasal structures that are problematic for formal and computational representation by

L_1. I have also provided rendering rules, in a recursive parallel with the syntax, which render phrases, up to sentences, into expressions of L_{IL} for semantical representation. The choice of fragment was to a large extent arbitrary — the idea is that no matter which basic expressions we choose to work with, it should be possible to provide an analysis analogous to that of the fraction of HL used in this essay. It is clear, however, that this is not an exhaustive theory of how HL works.

Furthermore, I have discussed CFGs and how they relate to HL. We have seen how they can be used to provide computational theories of HL grammar. For future work, however, they should be given more thorough linguistic motivations and be introduced in a more computational manner. We have also seen some ways in which CFGs are inadequate in that they need a large amount of rules and auxiliary syntactic categories. These auxiliary categories are often very artificial and do not represent intuitive features of actual HL. We have also seen how tree structures corresponding to the CF rules can be added to improve (and extend) a given CFG. These are important for new approaches to the construction of a syntax-semantics interface for HL.

Since Montague [12] much research has focused on improving his theory. One obvious way to improve our Montagovian grammar AGr is to extend it so that it reflects a larger amount of linguistic features in English language. The approach covered in this essay comes a long way, but there are many features that are left unexplained. For example, we have no way of handling verb agreement. There is no mechanism that reflects the fact that "Jacques smokes and sings" is an acceptable sentences whereas "Jacques smoke and sings" is not. Another limitation is its inability to deal with passive sentences. As of now, we are very capable of providing a satisfactory rendering of "Martin Heidegger wrote a silly book". However, there it is not at all obvious how the passive form of the sentence "A silly book was written by Martin Heidegger" could be analysed in a way that captures the fact that both sentences share the exact same truth conditions. We also have some overgeneration, especially when it comes to the negation of sentences and adjectives.

A second improvment is to make Montague's theory simpler and more computational. One important and relatively early attempt is Musken's logic TT_2, see Muskens [14]. Another more recent improvement is Moschovakis's theory of acyclic recursion, see Moschovakis [13].

To summarise, I need to:

1. linguistically motivate the use of CFGs and introduce the CFGs in a more computational way
2. find a more suitable computational grammar for the syntax of HL to reduce some of the complexity introduced by AGr and to be able to handle more features of HL (by a computational syntax-semantics interface)
3. solve problems with overgeneration
4. learn more about (i) computational syntax of HL, (ii) problems relating to semantics and (iii) better logics than L_{IL}

References

[1] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation, and Compiling*. 1,2 vols. Prentice-Hall series in automatic computation. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1972.

[2] Donald Davidson. "Truth and Meaning". In: *Synthese* 17.3 (1967), pp. 304–323. ISSN: 00397857, 15730964. URL: http://www.jstor.org/stable/20114563.

[3] David R Dowty, Robert Wall, and Stanley Peters. *Introduction to Montague semantics*. Vol. 11. Springer, 1981. URL: http://link.springer.com/book/10.1007%2F978-94-009-9065-4.

[4] James Garson. "Modal Logic". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2016. Metaphysics Research Lab, Stanford University, 2016.

[5] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. 3. ed., New international ed. Harlow: Pearson Addison-Wesley, 2014. ISBN: 1-292-03905-1.

[6] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. 1979.

[7] Rodney Huddleston, Geoffrey K Pullum, et al. *The Cambridge Grammar of English*. Cambridge University Press, 2002. URL: http://www.cambridge.org/uk/linguistics/cgel/.

[8] Jong-Bok Kim and Peter Sells. *English Syntax: An Introduction*. CSLI Lecture Notes. CSLI Publications, 2008.

[9] Roussanka Loukanova. "Muskens' Relational Montague Grammar. Part III: PTQ Revised". Computational Semantics II. Lecture Notes: Presentations. 2008.

[10] Roussanka Loukanova. "An Approach to Functional Formal Models of Constraint-Based Lexicalized Grammar (CBLG)". In: *Fundamenta Informaticae* 152.4 (2017), pp. 341–372. ISSN: 0169-2968 (P) 1875-8681 (E). DOI: 10.3233/FI-2017-1524.

[11] Roussanka Loukanova. "Partiality, Underspecification, Parameters and Natural Language". In: *Partiality and Underspecification in Information, Languages, and Knowledge*. Ed. by Henning Christiansen et al. Cambridge Scholars Publishing, 2017, pp. 109–150. URL: http://www.cambridgescholars.com/partiality-and-underspecification-in-information-languages-and-knowledge.

[12] Richard Montague. "The Proper Treatment of Quantification in Ordinary English". In: *Approaches to Natural Language*. Ed. by Patrick Suppes, Julius Moravcsik, and Jaakko Hintikka. Dordrecht, 1973, pp. 221–242.

[13] Yiannis N. Moschovakis. "A Logical Calculus of Meaning and Synonymy". In: *Linguistics and Philosophy* 29.1 (Feb. 2006), pp. 27–89. ISSN: 1573-0549. DOI: 10.1007/s10988-005-6920-7. URL: http://dx.doi.org/10.1007/s10988-005-6920-7.

[14] Reinhard Muskens. *Meaning and Partiality*. Studies in Logic, Language and Information. Stanford, California: CSLI Publications, 1995.

[15] Bertrand Russell. "On Denoting". In: *Mind* 14.56 (1905), pp. 479–493. ISSN: 00264423, 14602113. URL: http://www.jstor.org/stable/2248381.

[16] Ivan A. Sag, Thomas Wasow, and Emily M. Bender. *Syntactic Theory: A Formal Introduction*. Stanford, California: CSLI Publications, 2003.

[17] Alfred Tarski. "The Semantic Conception of Truth: and the Foundations of Semantics". In: *Philosophy and Phenomenological Research* 4.3 (1944), pp. 341–376. ISSN: 00318205. URL: http://www.jstor.org/stable/2102968.

[18] Dag Westerståhl. "Logical Constants in Quantifier Languages". In: *Linguistics and Philosophy* 8.4 (1985), pp. 387–413. ISSN: 01650157, 15730549. URL: http://www.jstor.org/stable/25001218.

[19] Dag Westerståhl. "Formal Semantics 4: Proper Names, Coordination, Type Shifts for the Connectives". Formal Semantics. Lecture notes: Presentations. 2016.

[20] Yoad Winter. "Syncategorematic conjunction and structured meanings". In: *In Proceedings of Semantics and Linguistic Theory, SALT5*. 1995.