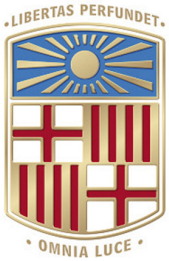
University Of Barcelona
Barcelona, Catalonia, Spain
2024



Final Master's Project

Lexical Semantics in Modern Type Theory: The challenge of selectional coercion

Author:
Stepan G. Kuznetsov

Advisors:
Peter R. Sutton & **Joost J. Joosten**

Faculty of Philosophy,
Master in Pure and Applied Logic

Contents

to Varya P.

I'm scared that at a glance
upon two equal things
I do not notice they are different,
that each lives only once.
I'm scared that at a glance
upon two equal things
I do not see they're eagerly
trying to be alike.

A. Vvedensky
(translated by V. Pechorin)

1 Introduction

1.1 Outline

Modern Type Theory (MTT) is a framework used for natural language semantics. It is built upon "richly-typed theories" introduced by P. Martin-Löf (Martin-Löf 1998) and applied to formal semantics for the first time by G. Sundholm (Sundholm 1986) and U. Mönnich (Mönnich 1985) with further studies by A. Ranta (Ranta 1995). The system considered in this work follows Ranta's ideas of applying Martin-Löf type theory to natural language semantics and was studied and developed by Z. Luo in his own work and through multiple collaborations with others (e.g. Luo 1994, Luo, Soloviev, and Xue 2013, Chatzikyriakidis and Luo 2020). The main difference of MTTs from the classic Montagovian approach (Montague 1970, Gallin 1975) is that MTTs are based on the *many-sorted* logical system as opposed to the *single-sorted* simple Church type theory where only two basic types exist. Richly-typed theories which possess more base types along with dependent and inductive types allow one to analyze phenomena which are hard or even impossible to model in the Montagovian framework.

Moreover, with presence of propositional types the system is ensured with the internal logic which, with use of the Curry-Howard principle of "propositions as types" (Curry and Feys 1958, Howard 1980), supplies MTTs with proof-theoretic semantics. This is very important for the system's decidability and hence the possibility to use proof assistants (e.g. Coq 2010) to implement automation. Proof-theoretic semantics can be considered as the incorporation of system's semantics into the syntax itself which is a big advantage over Montagovian purely set-theoretic semantics.

MTT's rich type system is many sorted (i.e. having many base types) and has inductive types and propositional types. A lot of new tools such as coercive subtyping and dot types were developed to supply this system with strong proof-theoretic semantics and ability to model a vast variety of linguistic phenomena, with one of the notable ones being a copredication over polysemous words and quantification over them.

The research presented in my thesis aims to enrich the system described in Chatzikyriakidis and Luo 2020 with adoption of several ideas from the Generative Lexicon developed

by J. Pustejovsky (e.g. Pustejovsky 1996) in order to formalize the use of lexical information in meaning-derivations. In the Generative Lexicon, the lexical information of words is organized into special records, or "meta-entries", and word meanings can be coerced (i.e. shifted) via several compositional mechanisms to their lexical *meta-properties* thus committing to the induction of the phrase meaning in a finer and at the same time more restricted way.

1.2 Our Proposal

Linguistic coercion is a process under which the direct meaning of a word shifts to another meaning. Sometimes this meaning is fixed by the context and sometimes one of the many meanings inherent to the word gets chosen: When we say "Mary is a fast driver" we mean "Mary drives fast" (or, "Mary is a person who drives fast") - here as part of the meaning derivation, there is arguably a coercion that shifts the noun "driver" (which denotes humans) to the event of driving inherent to the noun, such that this event can be modified by *fast*; When we say "John picked up and read a newspaper" the newspaper should be treated as both physical object (which is getting picked up) and informational object (which is getting read), here coercion shifts the type of the noun "newspaper" to a suitable connotation depending on requirements of each verb.

Pustejovsky (Pustejovsky 2008, section 1) describes how some linguistic coercions are governed by selectional mechanisms initiated by other words in the phrase, for example verbs and adjectives which apply to nouns and specify their meaning or adverbs which apply to verbs and specify them. These verbs, adjectives or adverbs are able to choose one of the meanings, *meta-properties*, inherently associated with the argument. These meta-properties are, for example, process of driving of a driver, process of reading of a book, baking of a cake, cutting of a knife. We call this type of inherent meaning shift of a word which is governed by another word in its surroundings *selectional coercion* and oppose it to *contextual coercion* where the inducted new meaning is hard-fixed by the specific contextual information of the phrase. Consider two examples:

(1) John started Ulysses;

(2) ? John started a cat.

Without any special context the phrase (1) is undoubtedly interpreted as "John started reading a book" (while still using lexical context of the word "book") while the phrase (2) cannot be plausibly analyzed at all unless the *special* context of the situation is given, such as, for instance: "John is making clay figures. Today he started a cat". The difference between coercions ocurring in the semantics of (1) and (2) is in the following: the interpretations of "Ulysses" is a book and books are *always* associated with the processes of reading them and therefore these processes can be used as the images the coercion maps to without any special contexts. On the contrary, ability to "start making a cat" belongs only to the special contextual use of the noun. By the use of *selectional coercion* in (1) we imply the process under which the verb "*start*", which requires an event as an argument, can choose one of the conforming *inherent* lexical roles that has the noun it acts on, thus shifting "start a book" into "start reading abook". In its turn, the *contextual coercion* happening in the phrase (2) also performs the meaning shift "a cat $\mapsto$ clay cat" which is only made salient by the context i.e. outside of the lexical definition of the noun "cat" itself.

The goal of this work is to address the challenge of modelling selectional coercions in frame of MTT semantics. We will claim that selectional coercions, but not contextual coercions can be modelled via incorporating notions of lexical records and the Lecial Conceptual Paradigm (LCP) from the Generative Lexicon (Pustejovsky 1996) (further: GL) thus drawing the line between them:

Main Questions: What are the possible ways of integrating lexical semantics from the Generative Lexicon for cases of linguistical coercion into Modern Type Theory? To what extent can selectional coercion be modelled in MTT enriched with GL-style lexical structure?

In order to address the main questions, we use methods and definitions which are already present in Modern Type Theories: namely, Σ-types (e.g. Luo 2021), coercive subtyping (Luo, Soloviev, and Xue 2013), unit-types (e.g. Luo 2011a, A) and dependent event types (Luo and Soloviev 2017). This introduces a number of challenges. We briefly describe these here and then break the Main Questions down into sub-questions (Q1)-(Q3).

First, the proposed sketch of integrating GL-style lexical structure into MMT-semantics in Luo 2011a turns out to be not in line with Luo's most recent work on verbal predicates and event semantics. We therefore need to find the best way to rework and integrate the main notion of that construction into compositional process of meaning derivation via use of dependent event types (Luo and Soloviev 2017) in the tradition of neo-Davidsonian event semantics (Davidson 1967, Parsons 1990):

(Q1) Are there any viable ways of modelling lexical records in MTTs with nested Σ-types via the use of event semantics?

Second, lexical coercion constructions and the differences between them create challenges for compositional semantics. For instance, it is non-trivial to account for the differences between *fast cat* and *fast driver* without breaking the consistency of the type system or defining unnecessary extensions:

(Q2) What viable ways are there of the use of the newly-defined lexical records for modelling selectional coercions in MTTs in automatic compositional manner without breaking consistency of the type system or defining unnecessary extensions?

Another question posed in this work is the possibility of allowing for some selectional coercions to the genuinely ambiguous. For instance, when the phrase "John started a book" is uttered, the listener cannot conclude if John started *reading* or *writing* a book without any given context. Even phrases like "Leo Tolstoy started War and Peace" may imply that Tolstoy started *reading* his own book (if he, for instance, decided to revisit his own work after many years had passed):

(Q3) Can the ambiguity readings of selectional coercion constructions be allowed without breaking the type theory?

Unfortunately, it will be shown in this work (Section 5.3.1), due to the coherence property of MTTs, no purely type-theoretical solution is available for parsing said ambiguity cases and in the "Further Research" Section (6) we sketch possible *meta-solutions* for this issue which involve changing the process of translating from syntactic structure to semantic one.

1.3 Structure of The Text

In order to provide sufficient background for addressing all the main questions posed in this work, in the subsequent two Sections (2 and 3) we give introduction to natural language semantics with MTTs in order for the reader to be familiar with the main notions and use possibilities of the framework:

- In Section 2 we will give a brief overview of Montagovian formal semantics for natural languages in order to provide a historical insight as well as to later highlight differences and merits of MTTs.
- Then, in Section 3, Modern Type Theories and their use and core advantages for formal semantics are described. Besides, all the formal inference rules are given in Subsection 3.2.

Section 4 describes the motivation behind adoption of Generative Lexicon for dealing with research aims raised in (Q1)-(Q3). We explain how cases of linguistic coercion were proposed to be modelled in MTTs and outline new ways to model it via Generative Lexicon insights. Also we list core notions of Generative Lexicon which we are considering useful to adopt in the proposal section of the thesis.

The main proposals and possible solutions we put forward are described in Section 5 together with a description of the limitations of the proposed solutions as well as problems which arise from adopting them.

In Section 6 a less formal solution is discussed as possible research for the future: In view of inability to model ambiguous meanings inside of the system itself we can look at the meaning formation at more abstract levels and define certain "metarules" performing selectional coercion.

In Section 7 we conclude.

2 Background: Simple Type Theories in Formal Semantics

In order to describe and formally define MTTs we start with an introduction to semantics in the tradition of Montague. A description of main ideas of the system gives a brief overview of the roots of formal semantics and also shows the original limitations with which Modern Type Theories successfully deal.

2.1 Montague Semantics

One of the first approaches to formalization of natural language semantics was proposed by Richard Montague in the 1970s (e.g. Montague 1970). The general idea is to use model theoretical semantics of the so-called Intensional Logic (IL) which acts as a bridge language between the syntax of a phrase and its meaning i.e. individuals in a set-theoretical model and relations between them.

The crucial concept which this approach follows is a notion of compositionality i.e. a process of constructing the meaning of a phrase using meanings of its constituents through a finite number of rules governing their combination. Having a combinatorial model of syntax we can build a compositional model of semantics mirroring each syntactic rule into a semantic one.

The Intensional Logic introduced by Montague is an extension of Church's Simple Type Theory with additional intensional operators[1] and it uses two basic types, one of them being a type of entities $\boldsymbol{e}$ and another one being a type of truth values $\boldsymbol{t}$. Moreover, it has as a pseudo-type $\boldsymbol{s}$ which is used to model itensionality: it is not a basic type as on its own it is not a type of the system and only constructions of form $\boldsymbol{s} \to \alpha$ are well formed for any other well formed type α.

But, illustrating the complexity of the original system, Barwise and Robin Cooper 1981 (p. 204) put it:

> Montague had a certain job that he wanted to do and used whatever tools he had at hand to do it. If the product looks a bit like a Rube Goldberg machine, well, at least it works pretty well.

Later it was shown in Gallin 1975 that the intensional logic IL can be translated into the two-sorted version of Russel's original type theory (Russell 1903) which Gallin denotes by TY_2. It is constructed in a more elegant and comprehensible way. There the intensional pseudo-type $\boldsymbol{s}$ is defined as a regular basic type and this definition greatly generalizes the system. Having $\boldsymbol{s}$ as a full-blown basic type sounds like making system stronger but later in Zimmermann 1989 it was proven that the fragment of TY_2 which is used as a translation of IL is actually translatable back to IL making these two systems equivalent in a certain sense[2].

2.2 Gallin's TY_2

Nowadays almost every researcher uses the TY_2 system when working with Montagovian-style formal semantics due to its approachability and equivalence to the original intensional logic framework. In this section we give a formal definition of TY_2 following handouts by Brasoveanu 2010 and Landman n.d. and give some linguistic examples.

Types: To compose a set TYPE containing types of the two-sorted type theory TY_2 we use a set of basic types BasTyp:

$$\mathrm{BasTyp} := \{\boldsymbol{e}, \boldsymbol{t}, \boldsymbol{s}\},$$

where:

- basic type $\boldsymbol{e}$ is a type of *entities* i.e. *individuals* which exist in some possible world at some point of time;
- basic type $\boldsymbol{t}$ is a type of *truth values*;
- basic type $\boldsymbol{s}$ stands for possible *world-time pairs*.

Then, all types of TY_2 are now constructed as the smallest set of strings such that:

- $\boldsymbol{e}, \boldsymbol{t}, \boldsymbol{s}$ are in TYPE;
- if α, β belong to TYPE then $(\alpha \to \beta)$ belong to TYPE as well.

[1] Apart from the quasi-type $\boldsymbol{s}$ used to define intensionality of an object the original Montague system also had operators $^\wedge$ and $^\vee$ but they are not needed in TY_2 system which we describe further in this text due to the lambda abstraction mechanism.

[2] Composition of the translation one way and the translation backwards does not yield an identity.

The arrow-types $\alpha \to \beta$ are associated with functions from objects of type α to objects of type β.

Expressions (terms): A countable set $\mathrm{CONST}_\alpha = \{c_1^\alpha, c_2^\alpha, \ldots\}$ denotes a set of constants for every type $\alpha \in \mathrm{TYPE}$.

A countable set $\mathrm{VAR}_\alpha = \{x_1^\alpha, x_2^\alpha, \ldots\}$ denotes a set of variables for every type $\alpha \in \mathrm{TYPE}$.

A set of expressions(terms) EXP_α for each type $\alpha \in \mathrm{TYPE}$ is defined inductively as follows:

- (Constants and variables) $\mathrm{CONST}_\alpha \cup \mathrm{VAR}_\alpha \subseteq \mathrm{EXP}_\alpha$ for any α in TYPE;
- (Application) if $\varphi \in \mathrm{EXP}_{\alpha\to\beta}$ and $\psi \in \mathrm{EXP}_\alpha$ then $\varphi(\psi) \in \mathrm{EXP}_\beta$;
- (λ-abstraction) if $x \in \mathrm{VAR}_\alpha$ and $\varphi \in \mathrm{EXP}_\beta$ then $\lambda x.\varphi \in \mathrm{EXP}_{\alpha\to\beta}$;
- (Connectives) if $\varphi, \psi \in \mathrm{EXP}_{\boldsymbol{t}}$ then $\neg\varphi, \varphi \vee \psi, \varphi \wedge \psi, \varphi \to \psi \in \mathrm{EXP}_{\boldsymbol{t}}$;
- (Quantifiers) if $x \in \mathrm{VAR}_\alpha$ and $\varphi \in \mathrm{EXP}_{\boldsymbol{t}}$ then $\forall x\varphi, \exists x\varphi \in \mathrm{EXP}_{\boldsymbol{t}}$;
- (Identity) if $\varphi, \psi \in \mathrm{EXP}_\alpha$ then $(\varphi = \psi) \in \mathrm{EXP}_{\boldsymbol{t}}$.

Frames and Models: For each type $\alpha \in \mathrm{TYPE}$ the domain D_α is defined inductively as:

- $D_{\boldsymbol{e}} \neq \emptyset$;
- $D_{\boldsymbol{t}} = \{0, 1\}$;
- $D_{\boldsymbol{s}} \neq \emptyset$;
- $D_{\alpha\to\beta} = \{f | f : D_\alpha \to D_\beta\}$.

In the following, both for TY_2 and MTT, we will assume the right-associativity of arrow types i.e. $\alpha \to \beta \to \gamma = \alpha \to (\beta \to \gamma)$.

A *model* of TY_2 is a pair $\langle D, I\rangle$ where:

- A set D is a collection of all domains i.e. $D := \{D_\alpha\}_{\alpha\in\mathrm{TYPE}}$;
- An *interpretation function* I is map such that $I(c_\alpha) \in D_\alpha$ for any $c_\alpha \in \mathrm{CONST}_\alpha$.

Semantics: The semantics of TY_2 are defined in the usual way:

A function g is called an *assignment* if for any $\alpha \in \mathrm{TYPE}$ and every $x \in \mathrm{VAR}_\alpha$ we have $g(x) : D_\alpha$.

For any domain element $d \in D_\alpha$ and variable $x \in \mathrm{VAR}_\alpha$ a *substitution* $g[d/x]$ is a map g' such that:

$$g'(x) = d$$
$$g'(y) = g(y), \text{ if } y \neq x.$$

Having an expression A, a model $M = \langle D, I \rangle$ and an assignment g we denote the interpretation of A in M under g as $|A|^{M,g}$ or just $|A|$ and define it in the usual way:

(3) $|c| := I(c)$ if c is a constant;
$|x| := g(x)$ if x is a variable;

(4) $|\neg\varphi| := 1 - |\varphi|$; ('$-$' is the usual arithmetic substitution)
$|\varphi \wedge \psi| := \min(|\varphi|, |\psi|)$;
$|\varphi \vee \psi| := \max(|\varphi|, |\psi|)$;
$|\varphi \to \psi| := \max(1 - |\varphi|, |\psi|)$;

(5) $|\varphi(\psi)| := |\varphi|(|\psi|)$;

(6) $|\forall x_\alpha \varphi| := \min(|\varphi|^{g[d/x]})_{d \in D_\alpha}$;

(7) $|\lambda x_\alpha \varphi| :=$ function F with domain D_α such that
$F(d) := |\varphi|^{g[d/x]}$ for all $d \in D_\alpha$;

(8) $|\varphi = \psi| := 1$ if $|\varphi| = |\psi|$ and 0 otherwise.

Finally we define an entailment relation $\models$ as usual:

(9) For sets of TY_2-expressions Δ and Γ we say that Γ *entails* Δ if for every model M and every assignment g it holds that: $\min\{|\varphi|^{M,g}\}_{\varphi \in \Gamma} \leq \max\{|\psi|^{M,a}\}_{\psi \in \Delta}$ and it is denoted by $\Gamma \models \Delta$;

(10) A formula φ is said *valid* iff $\models \varphi$.

2.3 Intensionality

The most crucial (and the most complicated) part of Montagovian systems is their ability to work with intensionality. In modern studies this issue was first discussed by Frege (Frege 1892) where he used notions of "Sinn" and "Bedeutung" which can be translated as "sense" and "reference". Consider the famous example of the difference between "the morning star" and "the evening star". Both of these terms denote the planet Venus observable on the sky during different times of the day i.e. these terms have the same *extension* (i.e. *reference* or *bedeutung*) but their *intensions* are different. Therefore if terms a and b have coinciding references equalities of type $a = b$ do not always hold. Each sign has a designated reference but also it has a *sense* which is an embodiment of the mode of presentation. We know that "the morning star is seen in the sky near sunrise" means the same as "Venus is seen in the sky near sunrise" but the meaning of a phrase "George knows that the morning star is seen in the sky near sunrise" may be true with "George knows that Venus is seen in the sky near sunrise" being false. This distinction shows a necessity to always consider the *mode of presentation* of words as an additional layer on top of just an extension as intensionality largely affects the meaning derivation.

Later in time Rudolph Carnap (Carnap 1947), influenced by Wittgenstein's *Tractatus* (Wittgenstein 1922) developed a formal system which is able to model intensionality with use of possible world semantics. Quoting Fitting 2022:

> Carnap's fundamental idea is that intensions, for whatever entities are being considered, can be given a precise mathematical embodiment as functions on states, while extensions are relative to a single state. This has been further developed by subsequent researchers, of course with modern possible world

semantics added to the mix. The Carnap approach is not the only one around, but it does take us quite a bit of the way into the intensional thicket. Even though it does not get us all the way through, it will be the primary version considered here, since it is concrete, intuitive, and natural when it works.

As it was also suggested by Frege, there is a need to differ between *de dicto* and *de re* readings. Quine 1956 provides the following example:

Ralph believes someone is a spy.

The first possible reading is that Ralph believes that there is a certain person whom he keeps in mind and this person is a spy - this is called *de re* reading i.e. "about the thing". The second reading is that Ralph believes there is some spy in the world, without anyone particular in mind - this is *de dicto* or "about what is said". The difference here lies in the interpretation of the word "*someone*". Two readings can be expressed respectively like so:

(11) (*de re*) : $(\exists x)$(Ralph believes that x is a spy);

(12) (*de dicto*) : Ralph believes that $(\exists x)(x$ is a spy).

Another example of a similar nature is given in Landman n.d. Consider two sentences:

(13) Mary seeks an author of Ulysses;

(14) An author of Ulysses is an author of Finnegans Wake.

It is known that an extension of "*an author of Ulysses*" is James Joyce and it is true that it coincides with an extension of "*an author of Finnegans Wake*". But we cannot allow us to freely substitute these phrases with each other in all cases as in 15, which might have been a result of using equality expressed in 14 in the phrase 13:

(15) Mary seeks an author of Finnegans Wake.

The sentence in 15 is not the same as the phrase 13 if we interpret both of them *de re* as Mary might not know that these two books are written by the same author or she might not even know who the author of Ulysses is. In the case where Mary does not know the Ulysses's author, even the phrase *"Mary seeks James Joyce"* would have a meaning different from the meaning of 13. Considering this difference in interpretations, we can observe that the nature of verbs like *"believe"* or *"seek"* are intensional, not extensional (e.g., Montague 1970): extensions of pronouns or predicates might coincide but their intentions, and hence the possible meanings, can be different.

To deal with intensionality the third entity, (quasi-)type $\boldsymbol{s}$, is used in Montague's IL and in Gallin's TY_2. Objects of type $\boldsymbol{s} \to \alpha$ represent intensional maps i.e. correspondences between possible worlds or moments of time and assignments of phrase extensions. For example, for the case of the sentence 13, assume that Mary believes that it is Samuel Beckett who wrote Finnegans Wake. Then consider the map $AF(w, x)$ of type $\boldsymbol{s} \to (\boldsymbol{e} \to \boldsymbol{t})$ meaning "x is an author of Finnegans Wake at world w". During semantic analysis the predicate will be scoped to Mary's world w_M where $AF(w_M, x)$ will be true only when applied to the entity x corresponding to Beckett and not to Joyce, even though such an authorship is not true for the most of other worlds and for the objective reality we seem to live in. Having the notion of intensional functions the use of substitution performed in the sentence 15 is automatically restricted.

2.4 Examples

Even though there is no denial of the importance of intensionality, the scope of phenomena considered in this work is different. Then, further examples will have the base type $\boldsymbol{s}$ omitted[3] in order to bring attention to parts of TY_2 we are interested in. In order to analyze the sentence we need to assign types to the words in the chosen fragment of the language. A few basic examples (with type $\boldsymbol{s}$ omitted) you can see in the Table 1. There, syntactic categories CN, IV, ADJ, $\mathrm{ADJ}_{\mathrm{VP}}$ and Det follow the common notation from formal syntax (e.g. Chomsky 1957) and, informally speaking, they denote different parts of speech: common nouns, intransitive verbs, adjectives, adverbs and determiners respectively. The category S denotes a well formed sentence.

Syntactic Categories	**Example**	**Montague Semantics**
CN	man, cat	man, cat: $\boldsymbol{e} \to \boldsymbol{t}$
IV	walk, talk	walk, talk: $\boldsymbol{e} \to \boldsymbol{t}$
ADJ	handsome	handsome: $(\boldsymbol{e} \to \boldsymbol{t}) \to (\boldsymbol{e} \to \boldsymbol{t})$
ADJ_{VP}	quickly	quickly: $(\boldsymbol{e} \to \boldsymbol{t}) \to (\boldsymbol{e} \to \boldsymbol{t})$
Modified CN	handsome man	handsome(man): $(\boldsymbol{e} \to \boldsymbol{t})$
Det (Quantifier)	a, the, some	a, the, some: $(\boldsymbol{e} \to \boldsymbol{t}) \to (\boldsymbol{e} \to \boldsymbol{t}) \to \boldsymbol{t}$
S	a man talks	$\exists x : \boldsymbol{e}.man(x) \& talks(x) : \boldsymbol{t}$

Table 1: Typing examples for a simply typed semantics

These types convey the following meaning:

- Common nouns are typed as $\boldsymbol{e} \to \boldsymbol{t}$ because they are treated as predicates over entities i.e. if some entity a of the model is a cat the expression corresponding to this fact would be $cat(a)$. It would be of type $\boldsymbol{t}$ and actually equal to the value 1 (denoting truth) if in our model a is indeed a cat.

- The same goes for intransitive verbs, they are typed as $\boldsymbol{e} \to \boldsymbol{t}$ because they are treated as unary predicates displaying the state of some entity (or the absence of it). Therefore the fact "*John walks*" would be represented by an expression $walk(j)$, where j is a constant representing the person.

- Adjectives are defined as modifiers of common noun predicates in the sense that they are designed to be applied to nouns and result in expressions such as $handsome(person)$ which are of type $\boldsymbol{e} \to \boldsymbol{t}$. Thus, the sentence *"John is a handsome man"* would look like $handsome(man)(j)$ which is desirable.

- As for quantifiers, some of them can be defined through the logical language we already have. For example, the determiner *"a/an"* is typed as $(\boldsymbol{e} \to \boldsymbol{t}) \to (\boldsymbol{e} \to \boldsymbol{t}) \to \boldsymbol{t}$ and after applying a noun n and a predicate p to it the result would be an expression:

$$a(n, p) = \exists x(n(x) \wedge p(x)).$$

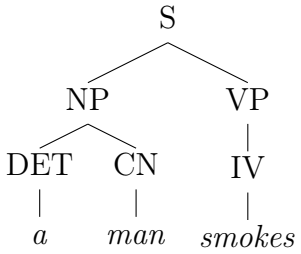
[3]Intensionality in MTTs is modeled in another way: with the use of many-sortedness and CN-as-types principle (3.1.1) $\boldsymbol{s}$ is not omitted but rather embedded into the basic types themselves: *Bachelor* and $Unmarried\_Man$ can have the same extension but they can be distinguished by their proofs for example. We will come back to it in more detail later.

In the case of the word *"every"*, the typing of the corresponding predicate would be the same but the computed result will use universal quantification instead:

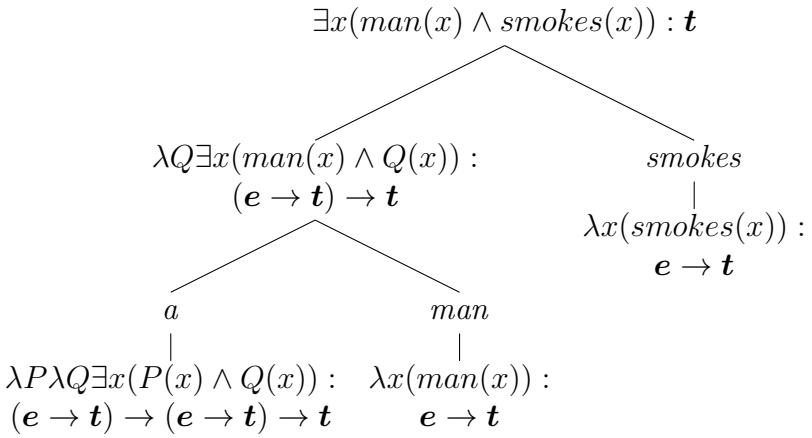
$$every(n, p) = \forall x(n(x) \to p(x)).$$

In order to analyze a natural language expression with Montague semantics we generate its syntactic structure (with use of tools whose definitions are out of the scope of this work). Then each word will become a leaf of a syntactic tree. Words (leaves) are then translated into the TY_2-typed objects corresponding to them. Each tree node, which is a rule application, is translated to its TY_2 counterpart. Then, after such transformation we apply rules to leaves in order to obtain a freshly-generated expression at the root and it will resemble the input phrase semantics.

For example, consider the sentence *"A man smokes"*. Its syntactic structure has the following shape:



Then after assigning a TY_2 type to each word and translating all the combinatorial syntactic rules we get a semantic tree with the desirable expression situated at the root:



2.5 TY_2 Limitations

(I) We can notice that definitions of most constituents and their behaviour in TY_2 are mirrored from formal syntax definitions. This is no coincidence because Montague's semantic system was designed in such a way that an expression describing the meaning of a sentence can be obtainable from the syntactic structure of it. Thus, constituent types and compositional rules in the semantics have to be somewhat similar to the syntactic ones. But while predicative nature of adjectives or verbs does not raise any questions, the predicative nature of common nouns causes several problems.

While syntactic systems are mostly proof-theoretic (e.g. Chomsky 1957, Lambek 1958), and therefore they have hope to be organized in a computationally decidable way, pure Montagovian approaches to formal semantics use IL or TY_2 only as intermediate

languages between the language syntax and set-theoretical models. This set-theoretical model side is what makes the computational side hazy and undecidable. Moreover, using TY_2 only as a bridge does not yield a very high level of linguistic restrictions as well as linguistic expressivity as opposed to the incorporation of more meaning into the logical language itself. Having a proof-theoretic semantics of the formal system used for the natural language analysis would make the formalization more rigorous as well as supply us with well-laid computational foundations e.g. an ability to model the system with proof-assistants such as Coq (Coq 2010).

(II) In TY_2 has only model-theoretic semantics: typing system exist separately from logical expressions and from the semantics of those expressions - that makes a system less transparent and requires more effort for its use. For example, a (generally absurd) sentence *"a table runs"* is syntactically correct and it corresponds to a well-formed TY_2-expression $\exists x(table(x) \wedge runs(x))$. This sentence cannot be judged as meaningful or meaningless (or as true or false) without considering possible set-theoretic models for it. Unless we provide a model in which *"table"* denotes a person with such a moniker, or in which the situation happens in a fictional reality where tables run, this sentence will be simply false, since the intersection between running things and tables is empty. However, we cannot explain why the sentence is semantically anomalous.

We can observe that in general context the noun *"table"* carries a special lexical meaning, it can be put only into contexts which require an object with a role of a piece of furniture. With a knowledge of a word's role we can conjecture immediately that *"a table runs"* is generally absurd. In the same way the noun *"cat"* which indeed can be treated as an entity which can run lets us easily conjecture that the phrase *"a cat runs"* in general would not cause any questions. These inherent roles are absent in standard TY_2 semantics due to its two-sorted type system. Following Chatzikyriakidis and Luo 2020 (p.23) we quote Link 1998 (preface):

> Language is able to refer to a wide array of objects of different sorts, which differ from each other in their characteristic structural properties. The universe of linguistic and philosophical discourse is thus most naturally taken as a multi-sorted domain containing all those objects.

Having a many-sorted system would simplify the process of language analysis as well as make it more meaningful. In the next chapter we look at such a system in the form of Modern Type Theory and see how it addresses the Montague system's issues mentioned in this section and how they broaden the expressivity of the meaning induction in natural languages.

3 Modern Type Theories in Formal Semantics: Overview

As said previously, natural language semantics in MTTs (Modern Type Theories) has a lot of common concepts adopted from the Montagovian tradition of formal semantics but, on the other hand, this framework has substantially departed and grown apart from the original view. Modern Type Theories are built upon the richly-typed theory approach of Martin-Löf (for example Martin-Löf 1984). The seminal application of such theories to natural languages was undertaken in Ranta 1995. Then in Luo 2011b (among others), following developments from Luo 1994 and Luo 1999 a version of the MTT system extended

with coercive subtyping was applied to formal semantics for the first time. In general, in this work we mostly follow the version of the MTT system described in the book by S. Chatzikyriakidis and Z. Luo (Chatzikyriakidis and Luo 2020).

First, we overview key merits of Modern type theories in Section 3.1. Then we give a formal definition of the calculus and its inference rules in Section 3.2 and in Section 3.3 we explain how natural language analysis is performed with use of MTTs giving some examples.

3.1 Core Additions of MTT

In this section, some of the main differences and advantages of MTTs compared to previously existent compositional systems are overviewed: CNs-as-Types principle, coercive subtyping, dependent and inductive types, dot-types, propositional types and proof-theoretic semantics, type universes and intensionality.

3.1.1 CNs as Types.

All common nouns in MTTs are interpreted as distinct types due to many-sortedness of the type theory. In TY_2 all nouns are defined in a coarse manner as predicates $\boldsymbol{e} \to \boldsymbol{t}$ from the universal domain of entities. In MTT, each common noun is interpreted (semantic interpretation is denoted by $[\![\ldots]\!]$) as its own unique type such as:

$$[\![human]\!] = Human : Type;$$
$$[\![man]\!] = Man : Type;$$
$$[\![book]\!] = Book : Type.$$

Many-sortedness brings more lexicality right into the type system itself and helps the establishment of proof-theoretic semantics: as each noun corresponds to the special type which represents it, the formal proof of the fact that an expression is typed as some noun (i.e. witnesses noun's type) already contributes to the truth statement inquiry of such assignment without a need to get into set-theoretic interpretations of it as we would do it with use of CNs-as-predicates. Compared to the Montagovian realization where the phrase "*John is a man*" is represented by the expression $man(John)$ the method of MTTs results in a *judgment* $John : Man$ which corresponds to the proof of this typing due to Curry-Howard propositions as proofs principle.

Moreover, with nouns interpreted as distinct types we can further constrain the typing system. In TY_2 the transitive verb "*kiss*" was typed as:

$$kiss : (\boldsymbol{e} \to \boldsymbol{t}) \to (\boldsymbol{e} \to \boldsymbol{t}) \to \boldsymbol{t}$$

and in MTT it can be typed as:

$$kiss : Human \to Human \to Prop.$$

We can see that the typing of "*kiss*" in MTT is more strict and it immediately disallows even a composition of expressions like $kiss(apple, book)$ (outside of a very special context) as it would produce a type clash because neither apple nor a book are humans. In order to find out the absurdity of the same expression in TY_2 we would have had to get one step deeper and examine set-theoretic models of it. In MTTs this process is unnecessary as the typing system does the checking for us earlier.

3.1.2 Coercive Subtyping

If we look at the previous example with the typing of the word "*kiss*" we can notice that the typing is $kiss : Human \to Human \to Prop$ and therefore we cannot directly form an expression $kiss(John, Mary)$ as $John : Man$ and $Mary : Woman$ and neither of those types is equal to $Human$. Therefore a system of subtyping relations is required in order to describe the fact that both men and women are indeed humans and can be used in contexts which require objects of type $Human$. Subtyping can be understood via an analogy to subset relations from set theory: it says that if the type A is known to be a subtype of the type B then any object a of type A can be regarded and hence used as an object of type B. Then, from both $Man \leq Human$ and $Woman \leq Human$ we can conclude that $kiss(John, Mary)$ is a well-typed expression as we can now deduce that $John : Human$ and $Mary : Human$.

The naive approach to formal definition of subtyping is to define a relation "$\leq$" straightforwadly (and reminiscent of a subset relation) with the "*subsumptive*" rule as follows:

$$\frac{a : A \quad A \leq B}{a : B}. \tag{16}$$

Unfortunately this does not work with MTTs due to its incompatibility with several mechanisms properties. One of the properties is called "*canonicity*" - it serves to endow the system with proof-theoretic clarity. Canonicity derives back to Martin-Löf studies but was first formulated later. It says that (Luo 2011b (p.40)):

Definition 1. ***Canonicity:*** *Any closed object of an inductive type is definitionally equal to a canonical object of that type.*

Canonical object, used in this definition is an element introduced via one of the introduction rules of the type. For example (Ranta 1995, pp.21-22), consider the definition of the type of natural numbers N (in a standard Peano style) with two constructors $0 : \mathrm{N}$ and $s : \mathrm{N} \to \mathrm{N}$ together with the appropriate addition operation $+ : \mathrm{N} \to \mathrm{N} \to \mathrm{N}$. Then, non-canonical elements like $1, 2 : \mathrm{N}$ should be explicitly defined to be equal to corresponding canonical elements $1 = s(0) : \mathrm{N}$, $2 = s(s(0)) : \mathrm{N}$. Having canonical shapes of each element lets us use inductive definitions of operations such as $+$ defined only for canonical forms as follows:

$$\begin{cases} a + 0 = a : \mathrm{N} \\ a + s(b) = s(a + b) : \mathrm{N} \end{cases}$$

In the scope of subsumptive subtyping, it is not possible for the canonicity property to hold and this leads to inconsistency issues. The problem goes from the definition of inductive types which requires any object of inductive type be equal to some canonical object of this type and subsumptive subtyping allows one to regard same objects as different types. If an object a is of type A and A is a subtype of B then the object a *is* of type B as well - this is ambiguous and clearly rises problems for inductive definitions and canonicity. To illustrate this we can quote the following example from Xue 2013:

Example 1 (Xue 2013, 3.13)**.** *Consider an inductive type $List(M)$ consisting of objects of type M. Then would have the following subtyping rule:*

$$\frac{A < B}{List(A) < List(B)}$$

The type list has two constructors: $nil(A)$ for an empty list of type A and $cons_M(a, l)$ for an A-list l appended with the element $a : A$. If subsumption rule (16) is used we can derive:

$$\frac{nil(A) : List(A) \quad List(A) \leq List(B)}{nil(A) : List(B)}$$

But the object $nil(A)$ is not canonical for $List(B)$ as it neither is $nil(B)$ nor $cons(b, l)$.

A special treatment was developed to build subtyping relations in MTTs in order to preserve proof-theoretic properties: coercive subtyping (Luo 1997, Luo 1999). The idea of coercive subtyping implemented in MTT is that if A is a proper subtype of B (i.e. $A < B$) then there is an unique coercion (map) c from A to B such that an object a of type A can be freely used in any context $\mathcal{C}_B(\_)$ expecting the object of type B. Expression $\mathcal{C}_B(a)$ is well-typed and equal to $\mathcal{C}_B(c(a))$. This makes subtyping expressions only abbreviations as they do not state that objects of one type *are* objects of another one but rather mark that there exists a well-behaved map between them.

The judgment introduced for denoting the fact of a subtyping under coercion c is expressed as $A \leq_c B : Type$. Formal rules of application (CA) and definition (CD) are the following for any object f of type $(B)C$ (which means that it is an arrow type from B to C) (Luo, Soloviev, and Xue 2013 p.40):

$$\frac{\Gamma \vdash f : (B)C \quad \Gamma \vdash a : A \quad \Gamma \vdash A \leq_c B : Type}{\Gamma \vdash f(a) : C} \; CA$$

$$\frac{\Gamma \vdash f : (B)C \quad \Gamma \vdash a : A \quad \Gamma \vdash A \leq_c B : Type}{\Gamma \vdash f(a) = f(c(a)) : C} \; CD$$

What these expressions tell us is that:

(CA): if there is a function f which requires and object of type B as an argument to produce an object of type C and we have an object a of type A along with the judgment that A is a subtype of B under coercion map c then we can conclude that application of f a is well typed as an object of type C.

(CD): having the same function f, object a and a judgment about subtyping with the coercion c we can conclude that the object $c(a)$ (of the type C) can be freely substituted for the object a itself in the context of application to the function.

Now we can establish coercive subtyping relations as:

$$Man \leq_{c_1} Human, Woman \leq_{c_2} Human.$$

Then with use of new rules we analyze "*John kisses Mary*" as $kiss(John, Mary)$. It is well typed due to CA and we can omit coercions from $kiss(c_1(John), c_2(Mary))$ due to CD, the proof sketch is shown below in (17)-(18).

$$\frac{kiss : Human \to (Human \to Prop) \quad John : Man \quad Man \leq_{c_1} Human}{kiss(John) : Human \to Prop} \; CA \tag{17}$$

$$\frac{\dfrac{}{kiss(John) : Human \to Prop}\,17 \quad Mary : Woman \quad Woman \leq_{c_2} Human}{kiss(John, Mary) : Prop} \; CA \tag{18}$$

()

A notion that ensures that the system with coercive subtyping is well behaved is called *coherence*:

Definition 2 (Luo, Soloviev, and Xue 2013 Definition 2.2)**.** *The set of subtyping judgements (governed by a relation* LF_Δ*) is called coherent if:*

- *If* $\Gamma \vdash_\Delta A \leq_c B : Type$*, then* $\Gamma \vdash_\Delta A : Type$*,* $\Gamma \vdash_\Delta B : Type$ *and* $\Gamma \vdash_\Delta c : (A)B$*.*
- $\Gamma \nvdash_\Delta A \leq_c A$*, for any* Γ, A *and* c*.*
- *If* $\Gamma \vdash_\Delta A \leq_{c_1} B : Type$ *and* $\Gamma \vdash_\Delta A \leq_{c_2} B : Type$*, then* $\Gamma \vdash_\Delta c_1 = c_2 : (A)B$*.*

It was proven that under the coherency of the set of subtyping relations the extension of the type system with this set of coercive subtyping relations is a conservative extension of it (Luo, Soloviev, and Xue 2013 Theorem 3.9) and therefore the consistency of the system is maintained.

3.1.3 Dependent Types and Inductive Types

Apart from distinct types for common nouns rich typing in MTTs has several inductive types (Chatzikyriakidis and Luo 2020, 2.2). Here we concentrate on two of them: Σ-types and Π-types as the other ones (such as disjoint union types, finite types) are not used in this work. Dependent constructions of Σ and Π-types are generalizations of products and implications. They are used, for example, to model and restrict such linguistic processes as adjectival modification and quantification.

Π-types (Chatzikyriakidis and Luo 2020, 2.2.1) represent dependent functions. Each object of this type is of form $\Pi x : A.B(x)$ and it represents a λ-function f such that applied to the object a of type A it yields an object $f(a)$ of the dependent type $B(a)$. We can think of B as a family of types depending on the argument it gets, for example $Child(x)$ is a type of children of some human x for every $x : Human$. Then, the type $\Pi x : Human.Child(x)$ is a type of dependent functions from humans to their children.

In case of no dependency we get the common arrow type $A \to B$ from $\Pi x : A.B$.

Σ-types (Chatzikyriakidis and Luo 2020, 2.2.2) of dependent pairs are of the form $\Sigma x : A.B(x)$ where B is a family of types depending on the object. In the case when there is no type-dependence Σ-type acts as a product type. Σ-type consists of a pair (a, b) of objects of type A and $B(a)$ respectively along with two projections π_1 and π_2 such that $\pi_1(a, b) = a$, $\pi_2(a, b) = b$. Σ-types are used for adjectival modification as follows: let Man be a type of men and *handsome* a predicate of type $Man \to Prop$. Then to form the expression corresponding to the phrase "*a handsome man*" we use Σ-types and get the expression $\Sigma x : Man.handsome(x)$. Intuitively, objects of this Σ-type represent all the pairs (x, p) where x is of type Man and p is a proof of the fact that x is handsome.

Σ-types in MTTs exist along with the existential quantification which is defined for the system's internal logic (see ch. 3.1.5). Keeping both mechanisms, weak and strong sum types, makes the system more expressive as compared to Ranta's approach of using only Σ-types for modelling existential quantification.[4]

[4]For more detail about the merits of preserving both weak and strong sums see Luo 2021, Luo 2019.

3.1.4 Dot-Types

Some nouns are *inherently polysemous* i.e. they can have several different but inherent lexical roles corresponding to them. For instance, the word "*book*" is regarded as a physical object in the phrase "*to pick up a book*" or "*to throw a book*" and as an informational object in the phrase "*to master a book*". But at the same time it is regarded as both physical and informational object when we say "*to pick up and master a book*". Addressing both senses of one word at the same time is called *copredication* (Pustejovsky 1996).

To model polysemy in MTTs "*dot-types*" are used. Dot-type is defined as $A \bullet B$ and it is a subtype of both A and B. To form it we need its components and their supertypes to have an empty intersection (i.e. absence of objects types as both types at the same time) which corresponds to the definition of *inherent polysemy* from Pustejovsky 2005:

> <Inherent polysemy> is the ability to appear in selectional contexts that are contradictory in type specification.

Types like $Phys \bullet Info$ are well formed while types $Phys \bullet Phys$ or $Phys \bullet (Phys \bullet Info)$ are invalid. Proof-theoretic properties of the system will fail if we do not impose this orthogonality restriction on the supertypes as otherwise some expressions would be able to have several different proofs which is undesirable. If A is a subtype of $B \bullet C$ and both B and C have a common supertype D then we get two possible ways to coerce from object A to an object of D: one via coercing to B and then to D and another through coercing to C and then to D.

That said, as types $Phys$ and $Info$ do not intersect, we can define the type $Book$ as as subtype of $Phys \bullet Info$ and therefore from relations (19)-(21) we can obtain $Book \leq_{c_1 c} Phys$ and $Book \leq_{c_2 c} Info$.

(19) $Book \leq_c Phys \bullet Info$

(20) $Phys \bullet Info \leq_{c_1} Phys$

(21) $Phys \bullet Info \leq_{c_2} Info$

As $pick\_up$ and $master$ are typed as $Human \to Phys \to Prop$ and $Human \to Info \to Prop$ we can put $Book$ as a second argument to both of them due to coercions $c_1 c$ and $c_2 c$ and due to the subtyping rules.

3.1.5 Proof-theoretic Semantics and the Prop Universe

The applicability of the Curry-Howard principle (Curry and Feys 1958 and Howard 1980) and a formal definition in style of Gentzen sequent calculus (Ranta 1995) provides MTTs with proof-theoretic semantics which allows automatization of parsing in proof assistants such as Coq (Coq 2010). The proof-theoretic nature of MTTs transfers them from the position of intermediate language (as TY_2) to the fundamental language of phrase semantics.

A core merit of the proof-theoretic approach is the *propositions-as-types* principle. Martin-Löf 1984 (p. 7) states that:

> If we take seriously the idea that a proposition is defined by laying down how its canonical proofs are formed and accept that a set is defined by prescribing how its canonical elements are formed, then it is clear that it would only lead

> to unnecessary duplication to keep the notions of proposition and set (and the associated notions of proof of a proposition and element of a set) apart. Instead, we simply identify them, that is, treat them as one and the same notion. This is the formulae-as-types (propositions-as-sets) interpretation on which intuitionistic type theory is based.

MTT is built around *judgments* which are assertions about types, either about correctness of the formed type or about the typing assignment of some expression. Judgments have forms of:

$a : Man$,
$Book : Type$,
$sleeps(John) : Prop$

Then the whole system (for example LF_Δ in Chatzikyriakidis and Luo 2020) is organized as a calculus of judgments with a list of inference rules. A judgment is valid in the system if and only if there is a proof of it. Moreover, both LF and LF_Δ (which we discuss in detail further) also act as type-theoretic metalanguages for defining type-theoretic object languages. The proof-theoretic power of this approach allows us to reason about the constitution of the type theory that we are utilising for natural language analysis. For example the well-formedness of each type can be proven formally.

As every judgment represents an inhabitant of the type and, at the same time, as a correct judgment is identical to the presence of its proof, we can freely identify correct judgements with their proofs. This result is especially powerful in view of CNs-as-types principle, since the correct typing of an object representing a noun *is* the proof of it, and hence we do not have to go out to the model-theory side (which is also present in MTTs) to check if some assertion is correct.

Logical propositions can be formed in MTTs as objects of type $Prop$, which is a *"logical universe"*, - "an internal totality in the type theory" (Chatzikyriakidis and Luo 2020 p.34). Due to impredicative nature of the $Prop$ universe (as every proposition is a type) we can quantify over $Prop$ itself and therefore define all the usual logical operations, existential quantifier and λ-abstraction in terms of universal quantification, the introduction rule for which is (Chatzikyriakidis and Luo 2020, A3.1):

$$\frac{\Gamma \vdash_\Delta A : Type \quad \Gamma, x : A \vdash_\Delta P : Prop}{\Gamma \vdash_\Delta \forall x : A.P : Prop}$$

For example, the phrase "*if a man is handsome he is not ugly*" is represented by (24) with use of two predicates as follows (Chatzikyriakidis and Luo 2020, p.26):

(22) $handsome : Human \rightarrow Prop$
(23) $ugly : Human \rightarrow Prop$
(24) $\forall x : Man.(\forall p : handsome(x).\neg ugly(x)) : Prop.$

As the topmost clause the quantifier argument (which is $handsome(x)$) is typed as $Prop$ and it does not appear free in the second part we can impose a notation $\Rightarrow$ to reform an expression in a more concise manner (Chatzikyriakidis and Luo 2020 A3.2):

(25) $P \Rightarrow Q := \forall x : P.Q$
(26) $(24) = \forall x : Man(handsome(x) \Rightarrow \neg ugly(x)) : Prop.$

3.1.6 Type universes:

A Type universe is a way to denote some collection of types in order to quantify over them. Following Martin-Löf 1998 we informally define universes as:

> A type V which will be called a universe and whose objects are to be types, together with the reflection principle which roughly speaking says that whatever we are used to doing with types can be done inside the universe V.

When combined with Π-type constructors, type universes allow representation of various *polymorphic* concepts. One of the universes used in MTTs, CN, denotes a universe of all (representations of) common nouns (Chatzikyriakidis and Luo 2020, Section 2.3.2). It is implemented as a type to which all noun types belong. For example $Human$, a type of all humans, belongs to CN - this is represented by the judgment $Human : CN$. Now we can use Π types to define polymorphic verbal modifiers like "*quickly*" and determiners like "*some*" or "*a*" as follows:

$$quickly : \Pi A : CN.(A \to Prop) \to Prop;$$
$$a, some : \Pi A : CN.(A \to Prop) \to Prop.$$

Here the application of the first argument (which is a type of CN universe), binds the rest to it. This way the phrase "*a man shouts*" is processed as follows:

1. determiner "*a*" is typed as above and has the form:

   $$[\![a]\!] = \lambda A : CN.\lambda P : (A \to Prop).(\exists x : A.P(x)).$$

   Applying it to the word "*man*" (which is of type Man) parameterizes the rest of the expression resulting in $a(man) : (Man \to Prop) \to Prop$.

2. the verb "*shouts*" is typed as $shouts : Human \to Prop$. As Man is a subtype of $Human$ we can pass the verb as a second argument to $a(man)$ giving us the interpretation $a(man, shouts) = \exists x : Man.shouts(x) : Prop$.

Another important universe is $LType$, a universe which was introduced by Chatzikyriakidis and Luo 2012 to address coordination, it contains all types suitable for coordination. After combining $LType$ with Π-polymorphism the interpretation of coordinating words such as "*and*" can be set as:

$$and : \Pi A : LType.A \to A \to A.$$

Now the object which represents "*and*" can be parametrized by any lexical type A and then would take a specified form of $A \to A \to A$.

Consider below interpretations of phrases "*John walks and Mary talks*" where coordinated parts are formed sentences typed as $Prop$ and "*John walks and talks*" where verbs are being coordinated at first before getting applied to the object (Chatzikyriakidis and Luo 2020 p.37):

$$[\![\text{John walks and Mary talks}]\!] = and(Prop, walk(j), talk(m));$$
$$[\![\text{John walks and talks}]\!] = and(Human \to Prop, walk, talk)(j).$$

3.1.7 Intensionality and Contexts

There is no "base type" $\boldsymbol{s}$ in MTTs as the one which was used in the original Montague semantics to model intensionality and belief states. Instead, rich typing allow us to define *contexts*. Contexts were proposed in Ranta 1995 as sequences of hypotheses of form:

$$x_1 : T_1(x_1), x_2 : T_2(x_1), \ldots, x_n : T_n(x_1, \ldots, x_{n-1}),$$

where $x_1 \ldots x_n$ are variables and $T_1, \ldots, T_n$ are types possibly depending on previously declared objects. This construction is used as a set of assumptions for conducting formal proofs of expressions i.e. notion of a linguistic context in MTTs coincide with the proof-theoretical notion of a context. In other words, from the formal point of view, contexts in MTTs are used as usual contexts from common formal logical systems with inference relations.

Ranta's definition of contexts can work only with typings of variables. In order to deal with the need to define constants or other constructions such as subtyping relations, *signatures* are used in MTTs (Luo 2014). A purpose of signatures is to describe situations (or incomplete possible worlds) which are related to the linguistic meaning of the word "context" as well. The notion of signature is introduced because there exist complications of embedding constants into contexts as contexts were initially designed only for variables. As Chatzikyriakidis and Luo 2020 (p.26) put it:

> However, to add these new forms of entries into contexts is not so easy: in particular, its meta-theoretic study is rather sophisticated and has been a difficult open question. Instead, it is easier to add them as entries in signatures.

The problem with possible worlds (and moments of time) from Montagovian approaches is the inability to enumerate them and hence to check them all. Here, as contexts and signatures are finite, provability can be decided in a finite time. As context and signature contain typing judgements and all the expressions (including nouns) are types we can encapsulate all the additional information we need within them.

Now the *de re* reading of the example of "*Mary seeks an author of Ulysses*" which we discussed on page 10 in Section 2.3 can be parsed properly because in the context of Mary's belief state there would be no record $Author\_Of(Joyce, Ulysses) : Prop$, where $Author\_Of(x, y)$ denotes an expression "*x is an author of y*". In the context (denoted by Γ) and signature (denoted by Δ) we would only have a records $Ulysses : Book$, $Author\_Of : Author \to Book \to Prop$ and $\exists x : Author.Author\_of(x, Ulysses)$. The whole expression with its linguistical context would be represented (rather informally) as:

$$\begin{aligned}
(context)\ \Gamma &= \exists x : Author.Author_of(x, Ullyses) : Prop;\\
(signature)\ \Delta &= Ulysses : Book, Author_of : Author \to Book \to Prop;\\
(judgment)\ \Gamma &\vdash_\Delta \exists x : (\Sigma y : Author.Author_of(y, Ulysses)).seeks(Mary, x) : Prop,
\end{aligned}$$

If Mary is not aware that it was James Joyce who wrote "Ulysses" or if she is not aware of the fact that he is also an author of "Finnegans Wake" then the phrase "*Mary seeks an author of Ulysses*" would not be provable as an equivalent to "*Mary seeks an author of Finnegans wake*" in this restricted situation.

3.2 Formal Definition: the system LF_Δ

Here we give the formal definition of the core parts of the system LF_Δ, following Chatzikyriakidis and Luo 2020 (A5.1). LF_Δ is an extension of the logical framework LF, developed

in the style of Martin-Löf type theory and defined in Luo 1994 (chapter 9). LF_Δ differs from LF in the presence of signatures in judgement forms and inference rules.

Both LF and LF_Δ use *kinds*, which act as types but they are types in the metalanguage defining type theories themselves. The metalanguage notion of kinds, while it does not substantially differ from the object language notion of types, is useful because it prohibits the combination of kinds and types in formal language expressions. Each object A of the special kind $Type$ is defined to correspond to a kind $El(A)$ which represents all the objects of the type A. Following Luo 1994 (p.169) when no confusion may occur we omit the notation $El(A)$ and use just A instead. As, for example in $a : (x : A)B$ in place of $a : (x : El(A)) : El(B)$ or $a : A$ instead of $a : El(A)$.

Judgements: judgements in LF_Δ (Chatzikyriakidis and Luo 2020, A5.1) can be of the following forms:

1. Δ *valid*, asserts that the signature Δ is valid;
2. $\vdash_\Delta \Gamma$, asserts that the context Γ is valid under the signature Δ;
3. $\Gamma \vdash_\Delta K$ *kind*, asserts that that K is a kind under the signature Δ in the context Γ;
4. $\Gamma \vdash_\Delta k : K$, asserts that k is an object of kind K in Γ under Δ;
5. $\Gamma \vdash_\Delta K_1 = K_2$, asserts that K_1 and K_2 are equal kinds in Γ under Δ;
6. $\Gamma \vdash_\Delta k_1 = k_2 : K$, asserts that k_1 and k_2 are equal objects of kind K in Γ under Δ.

Inference Rules: inference rules are given in this section. The empty sequence is denoted by $\langle\rangle$ and $dom(p_1 : K_1, \ldots, p_{:}K_n)$ denotes $p_1, \ldots, p_n$.

- Contexts and signatures (Chatzikyriakidis and Luo 2020, A5.1):

  The empty signature is well-formed; a new judgement can be added to the signature Δ if the constant name is distinct and its kind is well-formed under Δ; judgment from the context is valid under it:

  $$\frac{}{\langle\rangle \ valid} \qquad \frac{\langle\rangle \vdash_\Delta K \ kind \quad c \notin dom(\Delta)}{\Delta, c : K \ valid} \qquad \frac{\vdash_{\Delta, c:K, \Delta'} \Gamma}{\Gamma \vdash_{\Delta, c:K, \Delta'} c : K}$$

  The empty context is valid under a valid signature; a new judgement can be added to the context Γ under Δ if the variable name is distinct and its kind is well-formed in Γ under Δ; expressions from the context are valid in it:

  $$\frac{\Delta \ valid}{\vdash_\Delta \ \langle\rangle} \qquad \frac{\Gamma \vdash_\Delta K \ kind \quad x \notin dom(\Gamma)}{\vdash_\Delta \Gamma, x : K} \qquad \frac{\vdash_\Delta \Gamma, x : K, \Gamma'}{\Gamma, x : K, \Gamma' \vdash_\Delta x : K}$$

- Equality (Chatzikyriakidis and Luo 2020, A5.1):

  These rules represent reflexivity, commutativity and transitivity for both kind and object levels:

  $$\frac{\Gamma \vdash_\Delta K : \ kind}{\Gamma \vdash_\Delta K = K} \qquad \frac{\Gamma \vdash_\Delta K = K'}{\Gamma \vdash_\Delta K' = K} \qquad \frac{\Gamma \vdash_\Delta K = K' \quad \Gamma \vdash_\Delta K' = K''}{\Gamma \vdash_\Delta K = K''}$$

  $$\frac{\Gamma \vdash_\Delta k : K}{\Gamma \vdash_\Delta k = k : K} \qquad \frac{\Gamma \vdash_\Delta k = k' : K}{\Gamma \vdash_\Delta k' = k : K} \qquad \frac{\Gamma \vdash_\Delta k = k' : K \quad \Gamma \vdash_\Delta k' = k'' : K}{\Gamma \vdash_\Delta k = k'' : K}$$

An object of a kind K is an object of K' if these kinds are equal and the same about the object equality judgment of equal kinds:

$$\frac{\Gamma \vdash_\Delta k : K \quad \Gamma \vdash_\Delta K = K' \; kind}{\Gamma \vdash_\Delta k : K'} \qquad \frac{\Gamma \vdash_\Delta k = k' : K \quad \Gamma \vdash_\Delta K = K' \; kind}{\Gamma \vdash_\Delta k = k' : K'}$$

- The *Type* kind (Chatzikyriakidis and Luo 2020, A5.1):

  The kind *Type* is defined; for each type the kind of it's elements is defined; equal types have equal element kinds:

$$\frac{\Gamma \; valid}{\Gamma \vdash_\Delta Type \; kind} \qquad \frac{\Gamma \vdash_\Delta A : Type}{\Gamma \vdash_\Delta El(A) \; kind} \qquad \frac{\Gamma \vdash_\Delta A = B \; Type}{\Gamma \vdash_\Delta El(A) = El(B)}$$

- Dependent product kinds (Chatzikyriakidis and Luo 2020, A5.1): in LF_Δ instead of $\lambda x : K.b$ the notation $[x : K]b$ is used and $(x : K)K'$ is used in place of $\Pi x : K.K'$.

  Kind formation rule; kind equality rule; introduction rule; equality for objects rule:

$$\frac{\Gamma \vdash_\Delta K \; kind \quad \Gamma, x : K \vdash_\Delta K' \; kind}{\Gamma \vdash_\Delta (x : K)K' \; kind} \qquad \frac{\Gamma \vdash_\Delta K_1 = K_2 \quad \Gamma, x : K_1 \vdash_\Delta K_1' = K_2'}{\Gamma \vdash_\Delta (x : K_1)K_1' = (x : K_2)K_2'}$$

$$\frac{\Gamma, x : K \vdash_\Delta k : K'}{\Gamma \vdash_\Delta [x : K]k : (x : K)K'} \qquad \frac{\Gamma \vdash_\Delta K_1 = K_2 \quad \Gamma, x : K_1 \vdash_\Delta k_1' = k_2' : K}{\Gamma \vdash_\Delta [x : K_1]k_1' = [x : K_2]k_2' : (x : K_1)K}$$

  Dependent type application rule; equality for applications; application definitions:

$$\frac{\Gamma \vdash_\Delta f : (x : K)K' \quad \Gamma \vdash_\Delta k : K}{\Gamma \vdash_\Delta f(k) : [k/x]K'} \qquad \frac{\Gamma \vdash_\Delta f = f' : [x : K]K' \quad \Gamma \vdash_\Delta k_1 = k_2 : K}{\Gamma \vdash_\Delta f(k_1) = f'(k_2) : [k_1/x]K'}$$

$$\frac{\Gamma, x : K \vdash_\Delta k' : K \quad \Gamma \vdash_\Delta k : K}{\Gamma \vdash_\Delta ([x : K]k')(k) = [k/x]k' : [k/x]K'} \qquad \frac{\Gamma \vdash_\Delta f : (x : K)K' \; x \notin FV(f)}{\Gamma \vdash_\Delta [x : K]f(x) = f : (x : K)K'}$$

- Prop universe (Chatzikyriakidis and Luo 2020, A3.1):
  Inferences showing that *Prop* is a type; each object of type *Prop* is a type:

$$\frac{\vdash_\Delta \Gamma}{\Gamma \vdash_\Delta Prop : Type} \qquad \frac{\Gamma \vdash_\Delta P : Prop}{\Gamma \vdash_\Delta P : Type}$$

  Universal quantifier formation; universal quantifier introduction; application formation; application definition:

$$\frac{\Gamma \vdash_\Delta A : Type \quad \Gamma, x : A \vdash_\Delta P : Prop}{\Gamma \vdash_\Delta \forall x : A.P : Prop} \qquad \frac{\Gamma, x : A \vdash_\Delta b : P \quad \Gamma, x : A \vdash_\Delta P : Prop}{\Gamma \vdash_\Delta \lambda x : A.b : \forall x : A.P}$$

$$\frac{\Gamma \vdash_\Delta f : \forall x : A.P \quad \Gamma \vdash_\Delta a : A}{\Gamma \vdash_\Delta f(a) : [a/x]P} \qquad \frac{\Gamma, x : A \vdash_\Delta b : P \quad \Gamma \vdash_\Delta a : A}{\Gamma \vdash_\Delta (\lambda x : A.b)(a) = [a/x]b : [a/x]P}$$

Logic operators are defined via universal quantifier due to impredicativity of the system i.e. of the ability to quantify over the whole *Prop* universe:

$$\begin{aligned}
P \Rightarrow Q &:= \forall x : P.Q \\
\mathbf{true} &:= \forall X : Prop.X \Rightarrow X \\
\mathbf{false} &:= \forall X : Prop.X \\
P \wedge Q &:= \forall X : Prop.(P \Rightarrow Q \Rightarrow X) \Rightarrow X \\
P \vee Q &:= \forall X : Prop.(P \Rightarrow X) \Rightarrow (Q \Rightarrow X) \Rightarrow X \\
\neg P &:= P \Rightarrow \mathbf{false} \\
\exists x : A.P(x) &:= \forall X : Prop.(\forall x : A.(P(x) \Rightarrow X)) \Rightarrow X \\
(a =_A b) &:= \forall P : A \to Prop.P(a) \Rightarrow P(b)
\end{aligned}$$

- Subtyping and subkinding (Luo, Soloviev, and Xue 2013):

  Subkinding in signatures:

$$\frac{\vdash_\Delta A : Type \quad \vdash_\Delta B : Type \quad \vdash_\Delta c : (x : A)B}{\Delta, A <_c B \ valid} \qquad \frac{\vdash_{\Delta, A <_c B, \Delta'} \Gamma}{\Gamma \vdash_{\Delta, A <_c B, \Delta'} A <_c B : Type}$$

  Basic subkinding rule:

$$\frac{\Gamma \vdash_\Delta A <_c B : Type}{\Gamma \vdash_\Delta El(A) <_c El(B) : Type}$$

  Subkinding for dependent types:

$$\frac{\Gamma \vdash_\Delta K_1' = K_1 \quad \Gamma, x : K_1' \vdash_\Delta K_2 <_c K_2' \quad \Gamma, x : K_1' \vdash_\Delta K_2' \ kind}{\Gamma \vdash_\Delta (x : K_1)K_2 <_{[f:(x:K_1)K_2[x:K_1']]c(f(x))} (x : K_1')K_2'}$$

$$\frac{\Gamma \vdash_\Delta K_1' <_c K_1 \quad \Gamma, x : K_1' \vdash_\Delta [c(x)/x]K_2 = K_2' \quad \Gamma, x : K_1' \vdash_\Delta K_2' \ kind}{\Gamma \vdash_\Delta (x : K_1)K_2 <_{[f:(x:K_1)K_2[x:K_1']]f(c(x))} (x : K_1')K_2'}$$

$$\frac{\Gamma \vdash_\Delta K_1' <_c K_1 \quad \Gamma, x : K_1' \vdash_\Delta [c_1(x)/x]K_2 <_{c_2} K_2' \quad \Gamma, x : K_1' \vdash_\Delta K_2' \ kind}{\Gamma \vdash_\Delta (x : K_1)K_2 <_{[f:(x:K_1)K_2[x:K_1']]c_2(f(c_1(x)))} (x : K_1')K_2'}$$

  Structural subkinding rules:

$$\frac{\Gamma \vdash_\Delta K_1 <_c K_2 \quad \Gamma \vdash_\Delta K_1 = K_1' \quad \Gamma \vdash_\Delta K_2 = K_2' \quad \Gamma \vdash_\Delta c = c' : (K_1)K_2}{\Gamma \vdash_\Delta K_1' <_{c'} K_2'}$$

$$\frac{\Gamma \vdash_\Delta K <_c K' \quad \Gamma \vdash_\Delta K' <_{c'} K''}{\Gamma \vdash_\Delta K <_{c' \circ c} K''} \qquad \frac{\Gamma, x : K, \Gamma' \vdash_\Delta K_1 <_c K_2 \quad \Gamma \vdash_\Delta k : K}{\Gamma, [k/x]\Gamma' \vdash_\Delta [k/x]K_1 <_{[k/x]c} [k/x]K_2}$$

$$\frac{\Gamma, \Gamma' \vdash_\Delta K_1 <_c K_2 \quad \vdash_\Delta \Gamma, \Gamma''}{\Gamma, \Gamma'', \Gamma' \vdash_\Delta K_1 <_c K_2} \qquad \frac{\Gamma, x : K, \Gamma' \vdash_\Delta K_1 <_c K_2 \quad \Gamma \vdash_\Delta K = K'}{\Gamma, x : K', \Gamma' \vdash_\Delta K_1 <_c K_2}$$

Coercive application rules:

$$\frac{\Gamma \vdash_\Delta f : (x : K)K' \quad \Gamma \vdash_\Delta k_0 : K_0 \quad \Gamma \vdash_\Delta K_0 <_c K}{\Gamma \vdash_\Delta f(k_0) : [c(k_0)/x]K'} \ (\mathrm{CA}_1)$$

$$\frac{\Gamma \vdash_\Delta f = f' : (x : K)K' \quad \Gamma \vdash_\Delta k_0 = k_0' : K_0 \quad \Gamma \vdash_\Delta K_0 <_c K}{\Gamma \vdash_\Delta f(k_0) = f'(k_0') : [c(k_0)/x]K'} \ (\mathrm{CA}_2)$$

Coercive definition rule:

$$\frac{\Gamma \vdash_\Delta f : (x : K)K' \quad \Gamma \vdash_\Delta k_0 : K_0 \quad \Gamma \vdash_\Delta K_0 <_c K}{\Gamma \vdash_\Delta f(k_0) = f(c(k_0)) : [c(k_1)/x]K'} \ (\mathrm{CD})$$

- Dot-Types (Chatzikyriakidis and Luo 2020, A6):

Formation rule:

$$\frac{\vdash_\Delta \Gamma \quad \langle\rangle \vdash_\Delta A : Type \quad \langle\rangle \vdash_\Delta B : Type \quad \mathcal{C}(A) \cap \mathcal{C}(B) = \emptyset}{\Gamma \vdash_\Delta A \bullet B : Type} \ (\bullet)$$

where $\mathcal{C}(A)$ is a set of *components* defined as:

$$\mathcal{C}(T) = \begin{cases} \{T' | T \leq T'\}, \text{ if normal form of } T \text{ is not of form } X \bullet Y \\ \{X' | X \leq X'\} \cup \{Y' | Y \leq Y'\}, \text{ if normal form of } T \text{ is } X \bullet Y \end{cases}$$

Introduction rule:

$$\frac{\Gamma \vdash_\Delta a : A \quad \Gamma \vdash_\Delta b : B \quad \Gamma \vdash_\Delta A \bullet B : Type}{\Gamma \vdash_\Delta \langle a, b\rangle : A \bullet B} \ (\bullet\mathrm{I})$$

Elimination rules:

$$\frac{\Gamma \vdash_\Delta c : A \bullet B : Type}{\Gamma \vdash_\Delta p_1(c) : A} \ (\bullet\mathrm{E}_1) \qquad \frac{\Gamma \vdash_\Delta c : A \bullet B : Type}{\Gamma \vdash_\Delta p_2(c) : B} \ (\bullet\mathrm{E}_2)$$

Computation rules:

$$\frac{\Gamma \vdash_\Delta a : A \quad \vdash_\Delta \Gamma b : B \quad \Gamma \vdash_\Delta A \bullet B : Type}{\Gamma \vdash_\Delta p_1(\langle a, b\rangle) = a : A} \ (\bullet Comp_1)$$

$$\frac{\Gamma \vdash_\Delta a : A \quad \Gamma \vdash_\Delta b : B \quad \Gamma \vdash_\Delta A \bullet B : Type}{\Gamma \vdash_\Delta p_2(\langle a, b\rangle) = b : B} \ (\bullet Comp_2)$$

Projections as coercions:

$$\frac{\Gamma \vdash_\Delta A \bullet B : Type}{\Gamma \vdash_\Delta A \bullet B \leq_{p_1} A : Type} \ (\bullet Proj_1) \qquad \frac{\Gamma \vdash_\Delta A \bullet B : Type}{\Gamma \vdash_\Delta A \bullet B \leq_{p_2} B : Type} \ (\bullet Proj_2)$$

Coercion propagation:

$$\frac{\Gamma \vdash_\Delta A \bullet B : Type \quad \Gamma \vdash_\Delta A' \bullet B' : Type \quad \Gamma \vdash_\Delta A \leq_{c_1} A' \quad \Gamma \vdash_\Delta B = B' : Type}{\Gamma \vdash_\Delta A \bullet B \leq_{d_1[c_1]} A' \bullet B' : Type}$$

$$\frac{\Gamma \vdash_\Delta A \bullet B : Type \quad \Gamma \vdash_\Delta A' \bullet B' : Type \quad \Gamma \vdash_\Delta A = A' \quad \Gamma \vdash_\Delta B \leq_{c_2} B' : Type}{\Gamma \vdash_\Delta A \bullet B \leq_{d_2[c_2]} A' \bullet B' : Type}$$

$$\frac{\Gamma \vdash_\Delta A \bullet B : Type \quad \Gamma \vdash_\Delta A' \bullet B' : Type \quad \Gamma \vdash_\Delta A \leq_{c_1} A' \quad \Gamma \vdash_\Delta B' \leq_{c_2} B' : Type}{\Gamma \vdash_\Delta A \bullet B \leq_{d[c_1,c_2]} A' \bullet B' : Type}$$

where the coercions $d1$, $d2$ and d are defined as: $d1[c1](x) = \langle c1(p1(x)), p2(x)\rangle$, $d2[c2](x) = \langle p1(x), c2(p2(x))\rangle$ and $d[c1, c2](x) = \langle c1(p1(x)), c2(p2(x))\rangle$.

- Π and Σ types(Chatzikyriakidis and Luo 2020, A2.1 and A2.2): all inductive types (e.g. Π-types, Σ-types, disjoin union types, unit and finite types) which are used in the full system and described with inference rules can be defined formally through the mechanisms we already have but for convenience Π and Σ can be defined with their own constructors by in the following rules:
  - Π-types: formation; introduction; application; β-reduction:

$$\frac{\Gamma \vdash_\Delta A : Type \quad \Gamma, x : A \vdash_\Delta B : Type}{\Gamma \vdash_\Delta \Pi x : A.B : Type}\ (\Pi) \qquad \frac{\Gamma, x : A \vdash_\Delta b : B}{\Gamma \vdash_\Delta \lambda x : A.b : \Pi x : A.B}\ (Abs)$$

$$\frac{\Gamma \vdash_\Delta f : \Pi x : A.B \quad \Gamma \vdash_\Delta a : A}{\Gamma \vdash_\Delta f(a) : [a/x]B}\ (App) \qquad \frac{\Gamma, x : A \vdash_\Delta b : B \quad \Gamma \vdash_\Delta a : A}{\Gamma \vdash_\Delta (\lambda x : A.B)(a) = [a/x]b : [a/x]B}\ (\beta)$$

  - Σ-types: formation; introduction; projections (elimination rules); computation:

$$\frac{\Gamma \vdash_\Delta A : Type \quad \Gamma \vdash_\Delta B : Type}{\Gamma \vdash_\Delta \Sigma x : A.B : Type}\ (\Sigma)$$

$$\frac{\Gamma \vdash_\Delta a : A \quad \Gamma \vdash_\Delta b : [a/x]B \quad \Gamma, x : A \vdash_\Delta B : Type}{\Gamma \vdash_\Delta (a, b) : \Sigma x : A.B}\ (Pair)$$

$$\frac{\Gamma \vdash_\Delta p : \Sigma x : A.B}{\Gamma \vdash_\Delta \pi_1(p) : A}\ (Proj_1) \qquad \frac{\Gamma \vdash_\Delta p : \Sigma x : A.B}{\Gamma \vdash_\Delta \pi_2(p) : [\pi_1(p)/x]B}\ (Proj_2)$$

$$\frac{\Gamma \vdash_\Delta a : A \quad \Gamma \vdash_\Delta b : [a/x]B}{\Gamma \vdash_\Delta \pi_1(a, b) = a : A}\ (Conv_1) \qquad \frac{\Gamma \vdash_\Delta a : A \quad \Gamma \vdash_\Delta b : [a/x]B}{\Gamma \vdash_\Delta \pi_2(a, b) = b : [a/x]B}\ (Conv_2)$$

  - Subtyping for Σ and Π types (based on Luo 1999 (4.2)):

$$\frac{\Gamma \vdash_\Delta A \leq_c A' : Type \quad \Gamma, x : A \vdash_\Delta B(x) \leq_{c'[x]} B'(c(x)) : Type}{\Gamma \vdash_\Delta \Sigma x : A.B \leq_{d_\Sigma} \Sigma x : A'.B'}$$

    where $d_\Sigma(z) = (c(\pi_1(z)), c'[\pi_1(z)](\pi_2(z)))$ for $z : \Sigma x : A.B$.

$$\frac{\Gamma \vdash_\Delta A' \leq_c A : Type \quad \Gamma, x : A \vdash_\Delta B(x) \leq'_c [x]B'(c(x)) : Type}{\Gamma \vdash_\Delta \Pi x : A.B \leq_{d_\Pi} \Pi x : A'.B'}$$

    where $d_\Pi(f) = \lambda x : A'.c'[x](f(c(x)))$ for $f : \Pi x : A.B$
  - Other: LF_Δ has many other inductive types such as disjoint union types, unit and finite types and also it has manifest entries used to define equivalences. We will not cover them as they are not needed to address the questions we are concerned with in this work.
- *LType* universe (Chatzikyriakidis and Luo 2020, A4): essentially the *LType* universe that contains all conjoinable types and formally it is constructed of predicates

of form $\Pi x_1 : A_1 . \ldots \Pi x_n : A_n . Prop$ (denoted by $PType$), all the types representing CN and the universe CN itself:

$$\frac{}{PType : Type} \qquad \frac{}{Prop : PType} \qquad \frac{A : LType \quad P(x) : PType[x : A]}{\Pi x : A . P(x) : PType}$$

$$\frac{}{LType : Type} \qquad \frac{}{CN : LType} \qquad \frac{A : CN}{A : LType} \qquad \frac{A : PType}{A : LType}$$

3.3 Formal Semantics with MTTs

In this section we give some examples of natural language semantics with MTTs.

Here is the MTT version of the table 2.4 (p. 11) which showed typing examples in TY_2:

Syntactic Categories	**Example**	**MTT Semantics**
CN	man, cat	$Man, Cat : Type$
IV	talk	talk: $Human \to Prop$
ADJ	handsome	handsome: $Human \to Prop$
ADJ_{VP}	quickly	quickly: $\Pi A : CN.(A \to Prop) \to (A \to Prop)$
Modified CN	handsome man	handsome(man): $\Sigma x : Man.handsome(x) : Type$
Determiner (Quantifier)	a, the, some	a, the, some: $\Pi A : CN.(A \to Prop) \to Prop$
S	a man talks	$\exists x : Man.talks(x) : Prop$

Table 2: MTT-typing examples

3.3.1 CN-as-types in Action

Now each noun, instead of being a predicate, is treated as a distinct type. This allows other words to be typed more precisely: for example an intransitive verb "*talk*" now is typed not by $talk : (\boldsymbol{e} \to \boldsymbol{t})$ which allows any entity to be passed as an argument but rather as $talk : Human \to Prop$, specifying the exact type to be let in.

We can see that CNs-as-types principle already filters out some phrases which are meaningless in a general context a priori, as the phrase "*a table runs*" we mentioned before. Now the typing is the following:

$$[\![man]\!] := Man : Type$$
$$[\![table]\!] := Table : Type$$
$$[\![a]\!] := (\lambda T : CN . \lambda P : (T \to Prop).(\exists x : T.P(x))) : (\Pi A : CN.(A \to Prop) \to Prop)$$
$$[\![run]\!] := run : Anim \to Prop$$

where $Anim$ is a type of animated entities. As there is a subtyping relation $Man \leq Anim$ we can derive the expression $\exists x : Man.run(x)$ from the typing judgments listed above.

$$\begin{aligned}[\![a]\!](Man, run) &= \lambda T : CN . \lambda P : (T \to Prop).\exists x : T.P(x)(Man)(run) \\ &= (\lambda P : (Man \to Prop).\exists x : Man.P(x))(run) \\ &= \exists x : Man.run(x)\end{aligned}$$

The transition from the first line to the second can be done through the application rule of dependent product kinds. The transition from the second line to the third is done by the subtyping $Anim \to Prop \leq Man \to Prop$. The latter subtyping follows from $Anim \to Man$ and the rule of subtyping of dependent product kinds and the inference rule for subtyping application.

If we try to conduct a proof of the phrase "*a table runs*" then we would fail on the type clash of $run : Anim \to Prop$ and $P : Table \to Prop$ required by the object $a(table) : (Table \to Prop) \to Prop$ as there is no subtyping relation between types $Anim$ and $Table$ nor we have separately defined coercions between them.

Here you can see sketches of a proof of $runs(m) : Prop$ for m of type $Human$ and also a tree illustrating that the proof of $runs(t) : Prop$ is impossible for t of type $Table$:

$$\Gamma = \{t : Table, m : Man\}$$
$$\Delta = \{runs : Anim \to Prop, Human \leq_c Anim : Type\}$$

$$\frac{\dfrac{\vdash_\Delta \Gamma}{\Gamma \vdash_\Delta runs : Anim \to Prop} \quad \dfrac{\vdash_\Delta \Gamma}{\Gamma \vdash_\Delta m : Man} \quad \dfrac{\vdash_\Delta \Gamma}{\Gamma \vdash_\Delta Man \leq_c Anim}}{\Gamma \vdash_\Delta runs(m) : Prop}\ (\mathrm{CA}_1)$$

(27)
$$\frac{\Gamma \vdash_\Delta runs : Anim \to Prop \qquad \dfrac{\text{(no way to apply any rule)}}{(*)\ \Gamma \vdash_\Delta t : Anim}}{(*)\ \Gamma \vdash_\Delta runs(t) : Prop}$$

(28)
$$\frac{\dfrac{\vdash_\Delta \Gamma}{\Gamma \vdash_\Delta runs : Anim \to Prop} \quad \dfrac{\vdash_\Delta \Gamma}{\Gamma \vdash_\Delta t : Table} \quad \dfrac{\text{(no way to apply any rule)}}{(*)\ \Gamma \vdash_\Delta Table \leq'_c Anim}}{(*)\ \Gamma \vdash_\Delta runs(t) : Prop}\ (\mathrm{CA}_1)$$

We can see that the expression $\Gamma \vdash_\Delta t : Anim$ in the proof-tree (27) cannot be proven as t is fixed in the context to be of type $Table$. There is no subtyping of the type $Table$ to the type $Anim$ in the signature and hence we cannot apply coercion application rule in the proof tree (28) and these are the only possibille cases for the proof of a given judgement $\Gamma \vdash_\Delta runs(t) : Prop$.

3.3.2 Adjectival Modification

To model adjectival modification in Chatzikyriakidis and Luo 2020 (p. 65) adjectives are classified into *intersective, subsective, privative* and *non-committal* and each class is treated differently with the use of different mechanisms provided by MTT. Typings of each class of adjectives and examples are given in the Table 3.3.2.

Contrary to the Montagovian approach of typing adjectives as CN modifiers of type $(\boldsymbol{e} \to \boldsymbol{t}) \to (\boldsymbol{e} \to \boldsymbol{t})$, all adjectives in MTT are typed as $A \to Prop$ where $A : CN$. Their application is treated in various ways.

Classification	Inference	Example	MTT-types/mechanisms
Intersective	$Adj[N] \Rightarrow N\&Adj$	handsome man	Σ-types
Subsective	$Adj[N] \Rightarrow N$	large mouse	Π-polymorphism
Privative	$Adj[N] \Rightarrow \neg N$	fake gun	disjoint union types
Non-committal$_{VP}$	$Adj[N] \Rightarrow ?$	alleged criminal	modal collection

Table 3: A classification of adjectives (Chatzikyriakidis and Luo 2020, p.65, table 3.2)

Intersective adjectives:

Intersective adjectives are those which modify expression by intersecting their denotations. The quality delivered by an intersective adjective is not contingent on the type of the noun it modifies. For example the word "*black*" is an intersective adjective because a "*black cat*" is the entity which is both black and a cat (while a "former president" is neither former, nor a president). Other intersective adjectives are, for example, "handsome", "triangular", "English".

In order to treat intersective adjectives the alignment with Σ-types is used. According to Chatzikyriakidis and Luo 2020 (p. 67), such treatment was proposed in Mönnich 1985, Sundholm 1986 and Ranta 1995 but there the solutions were incomplete due to the absense of a fully sound subtyping mechanism. Here is the interpretation of "*black cat*" as done in MTTs with coercive subtyping:

$$\begin{aligned} &[\![black]\!] = black : Object \to Prop \\ &[\![cat]\!] = Cat : Type \\ &[\![black\ cat]\!] = \Sigma x : Cat.black(x) : Type \end{aligned} \tag{29}$$

Due to the CNs-as-types principle, we interpret modified nouns as types as well. Hence "*black cat*" is interpreted by the sigma type (29) and each object of this type would be a pair $(a, b) : \Sigma x : Cat.black(x)$ where a is of type Cat and b is a proof of $black(a)$ i.e. the fact that a is black.

This approach captures the behavior of intersective modifiers (i.e. the fact that we can always deduce both properties back separately as in *a black cat is both black and a cat*):

- Due to the rules governing projections of Σ-types (namely, the rule $Proj_1$ on Page 26) we can choose the first projection π_1 to be a coercion and it would result in the subtyping relation

  $$\Sigma x : Cat.black(x) \leq_{\pi_1} Cat$$

  meaning "*a black cat is a cat*";

- As for any adjective Adj an expression $Adj(x)$ is of type $Prop$ and every proposition is a proof, the construction $\Sigma x : CN.Adj(x)$ is a pair (x, p) of an object x and a proof of its modification p. Thus, if $\Sigma x : Cat.black(x)$ is derivable, there exists a proof of $black(x)$. Therefore, as $\pi_1(\Sigma x : Cat.black(x)) = x$ we have a proof of $black(\pi_1(\Sigma x : Cat.black(x)))$ meaning that "*a black cat is black*".

Notice that as the first projection of Σ-type is usually defined as a coercion the relation $\Sigma x : T.Prop \leq_{\pi_1} T$ still holds and therefore modified nouns still can be used in contexts

requiring these nouns. Setting aside the semantics of the definite article, consider the phrase "*(the) brown fox jumps*" (for some known $a : Fox$ in the context and simplified typings). The proof of it's correctness is in Figure 1 on the next page.

Moreover, as the first projection of Σ-types is a coercion, if $jump(a)$ is true in the model so would be $jump(\ (a, p)\ )$, where p is a proof that a is brown. It can be possible through using the projection $\pi_1(a, p) = a$ as a coercion. This fact is proven in Figure 2 below.

$\Gamma = \{a : Fox, p : brown(x)\}$
$\Delta = \{brown : Fox \to Prop, jump : Fox \to Prop, \Sigma x : Fox.Prop \leq_{\pi_1} F\}$

$$\frac{\Gamma \vdash_\Delta jump : Fox \to Prop \quad \Gamma \vdash_\Delta (a,p) : \Sigma x : Fox.brown(x) \quad \Gamma \vdash_\Delta \Sigma x : Fox.brown(x) \leq_{\pi_1} Fox}{\mathbf{\Gamma \vdash_\Delta jump(\ (a,p)\ ) : Prop}}\ (\mathrm{CA}_1)$$

Figure 1: Coercion of a modified noun

$$(\mathbf{D1}) \qquad \frac{\Gamma \vdash_\Delta jump : Fox \to Prop \quad \Gamma \vdash_\Delta (f,p) : \Sigma x : Fox.brown(x) \quad \Gamma \vdash_\Delta \Sigma x : Fox.brown(x) \leq_{\pi_1} Fox}{\Gamma \vdash_\Delta jump(\ (f,p\ ) = jump(\pi_1(f,p)) : Prop}\ (\mathrm{CD}_1)$$

$$(\mathbf{D2}) \qquad \frac{\Gamma \vdash_\Delta jump = jump : Fox \to Prop \qquad \dfrac{\Gamma \vdash_\Delta a : Fox \quad \Gamma \vdash_\Delta p : Brown(a)}{\Gamma \vdash_\Delta \pi_1(a,p) = a : Fox}\ (Conv_1)}{\Gamma \vdash_\Delta jump(\pi_1(a,p) = jump(a)) : Prop}$$

$$\frac{\dfrac{\mathbf{D1}}{\Gamma \vdash_\Delta jump(\ (f,p\ ) = jump(\pi_1(f,p)) : Prop}\ (\mathrm{CD}_1) \quad \dfrac{\mathbf{D2}}{\Gamma \vdash_\Delta jump(\pi_1(a,p) = jump(a)) : Prop}}{\mathbf{\Gamma \vdash_\Delta jump(\ (a,p)\ ) = jump(a) : Prop}}$$

Figure 2: Equality

Subsective adjectives

The second type of adjectives is "subsective". It includes such words as "*skilful*", "*large*", "*heavy*". All of them depend on the noun they modify as the large mouse is still small as an animal and some skilful typist is skilful as a typist, but not necessarily skillful with respect to other human activities (such as, say, cooking). To deal with this contingency the method of Π-polymorphism is used (Chatzikyriakidis and Luo 2013). If we just adopt the approach of using a plain Σ-type pair then some unnecessary relation would be derivable such as:

$$\Sigma x : Mouse.large(x) \leq \Sigma x.Animal.large(x)$$

which means that large mouse is a large animal (and that is never the case without a special context). What is done in order to avoid false generalizations is the following polymorhphic typing:

$$\Pi A : CN.(A \to Prop).$$

Now having $large : \Pi A : CN.(A \to Prop)$ we interpret "*large mouse*" with the same Σ-type pair but at first we parameterize the type of the adjective:

$$large : \Pi A : CN.(A \to Prop);$$
$$[\![mouse]\!] = Mouse : Type;$$
$$large(Mouse) : Mouse \to Prop.$$

Now we can form the expression

$$[\![large\ mouse]\!] = \Sigma x : Mouse.large(Mouse, x).$$

Now we do not have universal predicates $large$ or $skilful$ as all of them are at first parameterized by the type of the noun they modify and the large mouse will be large only as a mouse and not as an animal because the following subtyping won't hold:

$$(*)\ \ \Sigma x : Mouse.large(Mouse, x) \leq \Sigma x : Animal.large(Animal, x).$$

The subtyping does not hold as $large(Mouse, x) \not\leq large(Animal, x)$ because there is no subtyping relation $(Mouse \to Prop) \leq (Animal \to Prop)$.

Privative and non-cimmittal adjectives

Two remaining classes of adjectives are privative and non-committal. Their exact nature is disputable and a detailed discussion of them is outside the scope of this work. Below, we briefly and informally describe them.

Privative adjectives are words like "*fake*" or "*imaginary*". One of the ways to understand privative adjectives is to treat them as ones which negate CNs they modify: for example, as in "*a fake gun is not a real gun*". They are modelled in MTTs in Chatzikyriakidis and Luo 2020 (p.65) with use of *disjoint union types* that can model classes of both real and fake entities and later the presence of privative adjective reduces the typing to the "negated" part.

Non-committal adjective are the ones which does not rise any inference e.g. "*alleged*", "*potential*". An alleged murderer may be a murderer but may not be a murderer. What non-committal adjectives convey is a certain modality as all of them imply a presence of some person who poses a view: alleged murderer is the one who was alleged by someone. These adjectives are modelled in MTTs with use of collections of modalities $H_{h,\alpha}$ (defined with predicates of type $Prop \to Prop$) for each person h in the context and corresponding action α. When applied to propositions they represent what is being alleged, disputed, doubted etc. For example, "*John is an alleged murderer*" is analysed as the expression:

$$\exists h : Human.H_{h,alleged}(IS(Murderer, John)).$$

3.3.3 Dot-types for Co-predication

As briefly mentioned in Section 3.1.4, nominal polysemy is a phenomena of a noun having several orthogonal lexical roles and the case where both roles inherently correspond to one entity can be modeled in MTT with use of dot-types which allow us to parse sentences with *co-predication*. Consider the phrase "*John picked up and read a book*". Typing is the following:

$$\begin{aligned}
&John : Man;\\
&and : \Pi A : LType.A \to A \to A\\
&pick_up : Human \to Phys \to Prop;\\
&read : Human \to Info \to Prop;\\
&[\![book]\!] = Book : Type \quad (Book \leq_c Phys \bullet Info).
\end{aligned}$$

At first, due to Dot-types we can use $Book$ in both contexts which require objects of type $Phys$ and of type $Info$. Here is a sketch of the proof with the type names abbreviated to first letters for compactness:

$$\Gamma = \{b : B\}$$
$$\Delta = \{John : M, B \leq_c P \bullet I, pick\_up : H \to P \to Prop\}$$

$$\dfrac{\dfrac{\Gamma \vdash_\Delta B \leq_c P \bullet I \qquad \dfrac{\dfrac{\Gamma \vdash_\Delta P \bullet I : Type}{\Gamma \vdash_\Delta P \bullet I \leq_{p_1} P : Type}}{\Gamma \vdash_\Delta P \bullet I \leq_{p_1} P}}{\Gamma \vdash_\Delta B \leq_{c(p_1)} P} \qquad \Gamma \vdash_\Delta pick\_up(John) : P \to Prop \quad \Gamma \vdash_\Delta b : B}{pick\_up(John, b) : Prop}$$

As we can see from the typing of the word "*and*", the and-coordination in MTT is modeled with Π-polymorphism by some lexical type. This means that in order to form the proper expression both of the arguments of "*and*" have to be of the same type, while types of $read$ and $write$ are not the same - one requires $Phys$ and another one $Info$. Thanks to the subtyping relation $Book \leq_c Phys \bullet Info$ and $Phys \bullet Info \leq_{c_1} Phys$, $Phys \bullet Info \leq_{c_2} Info$, we can derive these subtypings of both of these verbs to have a

coinciding form:

$$
\begin{aligned}
&pick\_up: \\
&Human \to Phys \to Prop \leq Human \to Phys \bullet Info \to Prop \\
&\qquad\qquad\qquad\qquad \leq Human \to Book \to Prop; \\
&read: \\
&Human \to Info \to Prop \leq Human \to Phys \bullet Info \to Prop \\
&\qquad\qquad\qquad\qquad \leq Human \to Book \to Prop.
\end{aligned}
$$

The scheme of the proof here is the following:

$$
\dfrac{\Gamma \vdash_\Delta B \leq P \bullet I \qquad \dfrac{\dfrac{\Gamma \vdash_\Delta P \bullet I : Type}{\Gamma \vdash_\Delta P \bullet I \leq P : Type}\ (Proj_1) \qquad \Gamma \vdash_\Delta P \bullet I \leq P}{\Gamma \vdash_\Delta B \leq P}\ \begin{array}{l}\text{(introduction of } El() \text{ which we ommit)}\\ \text{(transitivity of subkinding)}\end{array}}{\dfrac{\Gamma \vdash_\Delta P \to Prop \leq B \to Prop \qquad \ldots}{\Gamma \vdash_\Delta H \to P \to Prop \leq H \to B \to Prop} \qquad \ldots}
$$

Now the whole phrase "*John picked up and read a book*" is represented by the following well-typed and derivable expression:

$$and(Human \to Book \to Prop, pick\_up, read)(John)(book) : Prop.$$

3.3.4 Unit Types for Type Overloading

Unit types $\mathbf{1}_w$ (Chatzikyriakidis and Luo 2020 A2.4, Luo 2011a) are defined for each word w and they represent singleton types i.e. types which have only one object. They can be defined with a use of inference rules as follows:

$$
\frac{\vdash_\Delta \Gamma}{\Gamma \vdash_\Delta \mathbf{1}_w : Type} \qquad \frac{\vdash_\Delta \Gamma}{\Gamma \vdash_\Delta w : \mathbf{1}_w}
$$

$$
\frac{\Gamma, z : \mathbf{1}_w \vdash_\Delta C(z) : Type \quad \Gamma \vdash_\Delta c : C(w) \quad \Gamma \vdash_\Delta z : \mathbf{1}_w}{\Gamma \vdash_\Delta \mathcal{E}_w(C, c, z) : C(z)}
$$

$$
\frac{\Gamma, z : \mathbf{1}_w \vdash_\Delta C(z) : Type \quad \Gamma \vdash_\Delta c : C(w)}{\Gamma \vdash_\Delta \mathcal{E}_w(C, c, w) = c : C(w)}
$$

Where two latter rules state that application of the elimination operator $\mathcal{E}_w$ computes to c when applied to the canonical object w.

In Luo 2011a it is explained how unit types can be used for dealing with some homonymous words in systems with coercive subtyping. Consider the following two phrases which are given in the article:

(30) John runs quickly

(31) John runs a bank

We see that the word "*run*" should be interpreted differently for those two phrases as it has two unrelated meanings:

$$[\![run]\!]_1 : Human \to Prop \qquad \text{(for (30))}$$
$$[\![run]\!]_2 : Human \to Insitution \to Prop \qquad \text{(for (31))}$$

In order to automatize the sense selection in the given context a special unit type can be defined and used as the way of modelling *type-overloading*:

$$[\![run]\!] : \mathbf{1}_{run}$$
$$\mathbf{1}_{run} \leq_{c_1} Human \to Prop$$
$$\mathbf{1}_{run} \leq_{c_2} Human \to Institution \to Prop$$

where:

$$c_i(run) = [\![run]\!]_i$$

Now the verb "*run*" is interpreted as an object *run* of type $\mathbf{1}_{run}$ and then, depending on the context, it is getting coerced to one of the designated typings. Note that in order to maintain coherence (Definition 2) the method of type overloading is available only when the word's typings are distinct and do not have a common supertype. For cases of words having distinct meaning but equal types other methods might be employed (for instance local coercions from Luo 2011a ch. 4).

3.3.5 Dependent Event Types

The notion of dependent types (Chatzikyriakidis and Luo 2020 (7.2), Luo and Soloviev 2017) derives back to studies of Davidson 1967 and later in the neo-Davidsonian tradition (e.g. Parsons 1990):

> Davidson has introduced events in semantics, claiming that verbs tacitly introduce existentially quantified events and that verbs and adverbial phrases are predicates over events. (Chatzikyriakidis and Luo 2020, p. 158)

With use of a distinct type *Event* phrases:

(32) John buttered the toast

(33) John buttered the toast with a knife in the kitchen

can be interpreted in the Montague-style system as (with j denoting John and t denoting a toast) the following respectively:

(34) $\exists v : Event.butter(v) \wedge (agent(v) = j) \wedge (patient(v) = t)$

(35) $\exists v : Event.butter(v) \wedge (agent(v) = j) \wedge (patient(v) = t) \wedge with\_knife(v)$
$\wedge in\_the\_kitchen(v)$

Here $agent(v)$ and $patient(v)$ map to objects which are agents and patiens of the event v. Agents ant patients denote two of many "thematic roles" used to describe event semantics. Agent is the one who conducts the action denoted by the predicate and patient is the one who undergoes the action. There are several other thematic roles such

as "instrument" and "theme" but in the current work we consider only agent and patient relations.

In MTTs, event types are presented in a more refined manner - as types parametrized by agents and patients. The system is extended with the following constant definitions (Luo and Soloviev 2017, 4.2):

$$Entity : Type,$$
$$Agent, Patient : Type,$$
$$Event : Type,$$
$$Evt_A : (Agent)Type,$$
$$Evt_P : (Patient)Type,$$
$$Evt_{AP} : (Agent)(Patient)Type.$$

And the natural subtyping rules are defined with a coherent set of parametrized coercions c_1, c_2, c_3, c_4 as follows:

$$Evt_{AP}(a,p) \leq_{c_1[a,p]} Evt_A(a) \qquad Evt_{AP}(a,p) \leq_{c_2[a,p]} Evt_P(p)$$
$$Evt_A(a) \leq_{c_3[a]} Event \qquad Evt_P(p) \leq_{c_4[p]} Event$$

The following extension is conservative and therefore well-behaved and it preserves all the properties of the original system.

4 Proposal: MTTs and Lexicality

In Section 4.1 we overview the way selectional coercion is modelled within MTTs, we pose some issues which arise from that solution, and we suggest how they may be confronted via adopting ideas from the Generative lexicon. Further, in Sections 4.2 and 4.3 we discuss notions of the Generative Lexicon and Lexical Conceptual Paradigm that we will later use in our proposal in Chapter 5.

4.1 Selectional Coercions in MTTs

As we discussed in Sections 3.1.4 and 3.3.3, dot-types were designed for so-called inherent polysemy i.e. an existence of several contradictory lexical roles (and hence typings) inherently associated with the word's meaning. Being interpreted by a dot-type the word can then be "coerced" to one of the type's constituents in the MTT framework sense (i.e. mapped to). For example a word "*book*" can be assigned with the dot-type $Phys \bullet Inf$ which is a pair of types representing a physical object and an informational object. Now both phrases *"thick book"* and *"interesting book"* are well typed as well as the phrase *"thick and interesting book"* due to subtyping relations $Phys \bullet Inf \leq Phys$ and $Phys \bullet Inf \leq Inf$ which are ensured by the dot-type definition. But note that these subtyping coercions of a dot type to its constituents are coercions only in the MTT sense i.e. they are maps with certain properties. These maps are usually not being linguistic coercions as both senses of the dot-typed word inherently belong to it, no shift of a meaning is happening.

In the literature, the above-mentioned polysemy where both meanings are inherent to the same noun representative has been differentiated from linguistic coercion, which is

also referred to as *selectional polysemy* (Pustejovsky 2008). Unlike (inherent) polysemy, linguistic coercion is analysed in terms of meaning shifts, not specified by the dot type. For instance, in the sentence *"she started a book"* where the sense of the word *"book"* is being shifted to the event of *"reading a book"*. The canonical MTT-solution proposed in Chatzikyriakidis and Luo 2020 is that there is a coercion map $reading : Book \to Event$ which maps objects of type $Book$ into objects of type $Event$ as they are required by the typing $start : Event \to Prop$. Given that there are many other possible coercions, we have a dictionary of such mapping rules that can be used during derivations. An approach of having a dictionary of coercions does not complicate noun definitions but neither does it quite correspond to the intuition of the real meaning derivation process in a strong way: some linguistic coercions are motivated by the nature of the interaction of the verb or adjective with the noun (as in (36) and (37) below), some are given from the context (as in (38)) and some depend on both compositional processes and a special given situations (as in (39)). Having all those different coercions modelled through coercive subtyping would mix their natures as well as declare that initial noun objects are *subtypes* of the entities they are being coerced to: while book is indeed a physical object and informational object it is not an event on itself, it is shifted to it as an exception. The same holds for a typist: someone who is a typist is not an event of typing, but it can be linguistically coerced to this event upon being in the proper context.

(36) "John is a skilful typist" $\mapsto$ "John types skilfully"

(37) "John started a book" $\mapsto$ "John started reading a book"

(38) "John ate the shell" $\mapsto$ "John ate the shell-shaped chocolate"

(39) "John started a cat" $\mapsto$ "John started making a clay cat".

Therefore, we will make a further distinction between coercions in the linguistic sense, namely, those that can be resolved in any out-of-the-blue context, e.g., (36), (37), and those that cannot, e.g., (38).[5] We call the first of these *selectional coercions*, a modelling of which is a main focus of this work stated on p. 5 of the Section 1.2.

Some coercions are *selectional coercions*, for instance, the ones which make shifts from a "*book*" to "*reading a book*" in (37) and "*typist*" to "*ability of typing*" in (36). Coercions of another type are context-dependent, *contextual*: in a situation where librarians are registering new books in a database of the library and say "*we started another book*" what is implied is not reading or writing but the registering of it.

Consider the following example given in Chatzikyriakidis and Luo 2020 (pp. 58-60) and in Asher and Luo 2013: let $Evt_A(a)$ be a type of events conducted by and agent a of type $Agent$. As the type of all humans, $Human$, is a subtype of $Agent$ the type $Evt_A(h)$ is well-formed for $h : Human$. Then typings for verbs can be defined accordingly as:

$$start, finish : \Pi h : Human.(Evt_A(h) \to Prop)$$
$$read, write : \Pi h : Human.(Book \to Evt_A(h))$$

Then in order to parse the expression "$Mary$ *started* "$War\ and\ Peace$"" which is interpreted as $start(Mary, W\&P)$ we need to coerce $W\&P$ (which is of type $Book$) to

[5]Note that the example in (39) combines both of the shift-senses we described as the coercion from "*a cat*" to "*a clay cat*" should be given in the context but the induction of "*to start making a clay cat*" from "*to start a clay cat*" is performed in a purely compositional manner without any more additions to the context.

the type $Evt_A(Mary)$ required by $start(Mary, \_)$. Coercion is defined with the following subtyping relation which utilizes *parameterized coercions*:

$$Book \leq_{c(h)} Event_A(h)$$

$$c(h,b) = \begin{cases} writing(h,b) & \text{if } h \text{ is an author of the book } b, \\ reading(h,b) & \text{otherwise.} \end{cases} \tag{40}$$

Here c is a family of coercions from type $Book$ depending on the agent of the action - if the agent is the writer of the book coercion application will result in a writing event and it will result in a reading event otherwise. Thus, the final and intended interpretation of the phrase in question is:

$$start(Mary, reading(Mary, W\&P)).$$

The parameterization of a coercion map is motivated by the coherence property of coercive subtyping: we cannot just allow two subtypings $Book \leq_{reading} Event$ and $Book \leq_{writing} Event$ as clearly $reading(b) \neq writing(b)$ but the equality is required by the coherence of subtyping judgements. That is why, in the scope of employing coercive subtyping to model linguistic coercions, the choice of the exact coercion image is needed. But it is not unnatural for sentences with coercion to be ambiguous: we might not know without a fixed context whether the person is starting to *read* or *write* a book. In this cases the coercion c stays unfolded as in:

$$[\![\text{Mary started a book}]\!] = \exists b : Book.start(Mary, c(Mary, b)).$$

As we see, modelling with coercive subtyping bears several formal complications. While we see that the judgement $Book \leq_c (h)Event$ allows us to successfully analyze the phenomena we want, we argue that books are not really subtypes of events. What actually happens is a full linguistic coercion, a real meaning shift to one of the inherent lexical roles of the word book and this shift is motivated by the interaction of the argument with the verb. If we could conjoin each noun with its lexical entry, we can draw fruitful linguistic generalizations from it. Notice that cases of phrases like *"she started a book"* and *"she started a cake"* have the same nature of a coercion process happening with objects *"cake"* and *"book"* in these contexts. The process of being read and the process of being baked are inherent contexts for a book and a cake respectively. Here, the use of a coercion dictionary would not reflect the common use of the verbs like *"start"* as verbs which often imply the use of lexical role of their objects, rather than the objects themselves.

Remember an example of contextual coercion we gave previously in this section about librarians registering books. In that case, the contextual coercion, as we call it, can be introduced into the logical context of the framework (in the case of MTTs into the signature Δ) when dealing with some particular case and it, in a certain sence, overloads the existing relation $Book \leq_{reading(h)} Evt_A(h)$ with the relation $Book \leq_{registering(h)} Evt_A(h)$. But we can see that *selectional* and *contextual* coercions seem to have different connections with nouns they shift: selectional coercion is applicable in every situation and it is getting triggered by the interaction with an appropriate function or modifier e.g. a verb or an adjective. Contextual coercion requires a special situation to occur as in the case with librarians, or the case of "the ham sandwich is sitting at Table 20" (Nunberg

1979) when the ham sandwich denotes a person whose order was this sandwich. There, we claim, the context redefines the role of the noun. But with the current realization in MTTs both kinds of coercions are not differentiated within the formal system as they are defined in the same way via coercive subtyping.

Thus, we might want to incorporate the description of a noun's behavior under selectional coercions into the noun definition instead of keeping the list of applicable coercion maps somewhere in our corpus. Then, such verbs as the ones described above, e.g. "to start", will be able to be defined as ones that commonly apply themselves to the lexical role of their objects (and hence performing a selectional coercion of these objects) rather then the object entities via using straightforward compositional manner. Explicit use of an external or contextual coercion then will be more visible as well as more constrained. It will be marked as an exceptional approach used only when more common processes fail or when the context has forcefully overloaded coercions.

Selectional coercions of a noun usually arise in the language from polysemous behaviour of other elements in the phrase, not from the mere polysemy of the noun. For example, polysemous adjective "*good*" can force the noun it modifies to coerce to one of it's properties in order to modify the property itself as in "*a good knife*" meaning "*a knife that is good (when used) for cutting*" or a "*a good driver*" meaning "*a driver who is good at driving*". Several compositional approaches were developed to model this behaviour. One of them is described in the framework called the Generative Lexicon (Pustejovsky 1996). There, with the use of the formalized lexical information, processes of coercion shifts are more visible as they are driven by precise selectional mechanisms: coercions are not taken from the detached dictionary but they are defined to reduce constituents to their lexical roles and they are parametrized accordingly. The result of the work of Pustejovsky is the notion of a system which meaningfully (w.r.t. the actual semantic interpretation) aligns formal definitions and the lexicon while modelling a vast amount of linguistic phenomena on a more precise level of detail.

The Generative Lexicon is an approach to formal semantics of natural languages proposed and extensively studied by J. Pustejovsky (e.g. Pustejovsky 1996). According to Pustejovsky 1996 (p.1) it addresses the following not-yet-studied issues of other systems:

- the creative use of words in novel contexts;
- an evaluation of lexical semantic models on the basis of compositionality.

The idea is to incorporate lexical information into a still rigorous and compositional formal model of the language meaning. This allows way more fine grained language analysis as an access to the lexical level yields a lot of knowledge which, if efficiently aligned and paired with compositionality, can be utilized elegantly without overwhelming the representation and usability. Among others, the generative lexicon allows to successfully model phenomena such as co-composition and copredication over polysemous words. Substantial part of the work on dealing with noun polysemy relies on the notion of Lexical Conceptual Paradigm: a collection of word's *modes of explanation* which is included into the definition of the word thus making its addressing linguistically and compositionally transparent.

An approach of adopting the notion of the *Lexical Conceptual Paradigm* (LCP) from

Pustejovsky 1996[6] might be suitable for an implementation inside of the framework of MTT's as Σ-types can let us model it. The final combination results in a richer level of detail in the analysis at the lexical level while maintaining the merits of MTT such as proof-theoretic semantics and decidability.

We now give an overview of Generative Lexicon in Sections 4.2, 4.3 and then we propose our own solution for emulationg GL notions in MTTs in order to model selectional coercions of a noun (Section 5).

4.2 Generative Lexicon: The Lexical Conceptual Paradigm

The Lexical Conceptual Paradigm was developed in order to describe a word behavior which depended on the context and thus involve more expressive mechanisms in the composition process. Incorporation of lexical roles into a single *meta-entry* associated with a word leads to a higher expressivity as well as higher and better constraints on the system and it effectively cuts the size of a lexicon.
Following Pustejovsky 1996 (ch 5), every lexical entry α is a labeled record and it consists of four components each one of them also being a labeled record:

$$\alpha = \langle \mathcal{A}, \mathcal{E}, \mathcal{Q}, \mathcal{I} \rangle.$$

- *Argument structure* $\mathcal{A}$ holds logical arguments of an entry;
- *Event structure* $\mathcal{E}$ is a definition of an event type;
- *Qualia structure* $\mathcal{Q}$ holds modes of explanation of an entry;
- *Lexical inheritance structure* $\mathcal{I}$ identifies of a relation of a current entry to others.

Qualia is the most important part of the use of the noun's lexical meta-entry as it stores the core information used for the parsing coersion processes as in phrases phrases like "*to start a book*". It represents an idea of a word definition through modes of explanation, the roles of an object which essentially define it. *Qualia structure* consists of four roles which together define a record. In the case of noun:

- *Constitutive*: a relation between an object and its parts or marks if it is a part of something else;
- *Formal*: that which distinguishes it whithin the larger domain;
- *Telic*: the purpose or a function;
- *Agentive*: factors or processes involved in its origin or "coming-into-being".

Constitutive (or Const): this field provides information about materials or some properties of the constituence of an object and as well marks if it is a part another entity. The first use is represented in the definition of a man, which is a male human and the second use is illustrated by the defition of the word "*hand*" which contains a fact that hand is a part of a body:

[6] first definition in Pustejovsky and Anick 1988

$$\left\{\begin{array}{ll}\underline{\boldsymbol{man}} & \\ Arg & = \left\{Arg1 \quad = x : human\right\} \\ Qualia & = \left\{\begin{array}{ll} Const & = male(x) \\ \ldots & \end{array}\right\} \\ \ldots & \end{array}\right\}$$

$$\left\{\begin{array}{ll}\underline{\boldsymbol{hand}} & \\ Arg & = \left\{Arg1 \quad = x : limb\right\} \\ Qualia & = \left\{\begin{array}{ll} Const & = part\_of(x, y : body) \\ \ldots & \end{array}\right\} \\ \ldots & \end{array}\right\}$$

Formal: If an object in the argument structure is simply typed (i.e. it is not a dot-object) then its formal quale mirrors the Arg field as the argument exhaustively defines the word. If the argument entry is dot-typed, then the formal field denotes a relation between logical entities which represent the word. Below you can see sketches of argument and formal fields of the words "*cake*" (unambigous) and "*book*" (polysemious) (Pustejovsky 1996 p.123, p.101 respectively):

$$\left\{\begin{array}{ll}\underline{\boldsymbol{cake}} & \\ Arg & = \left\{Arg1 \quad = x : Food\right\} \\ Qualia & = \left\{\begin{array}{ll} Formal & = x \\ \ldots & \end{array}\right\} \\ \ldots & \end{array}\right\}$$

$$\left\{\begin{array}{ll}\underline{\boldsymbol{book}} & \\ Arg & = \left\{\begin{array}{ll} Arg1 & = x : Phys \\ Arg2 & = y : Inf \end{array}\right\} \\ Qualia & = \left\{\begin{array}{ll} Formal & = hold(x, y) \\ \ldots & \end{array}\right\} \\ \ldots & \end{array}\right\}$$

In the same manner as a hand references a body which it is a part of, the record for noun *"body"* also references that its part is a hand.

Telic: this field defines a purpose of the noun. Pustejovsky 1996 (6.2.4) argues that it should encapsulate both *direct* and *purpose* usages i.e. telic shows either how one can act on an object or how this object can be used in order to commit other actions. An example of the former is a *"beer"* for which it is essential to be drunk and an example of the latter is a *"knife"* which is mainly used to cut:

$$\left\{\begin{array}{l}\underline{\textbf{\textit{beer}}} \\ Arg \quad = \left\{Arg1 \quad = x : liquid\right\} \\ Qualia \quad = \left\{\begin{array}{l} Telic \quad = drink(e,y,x) \\ \ldots \end{array}\right\} \\ \ldots \end{array}\right\}$$

$$\left\{\begin{array}{l}\underline{\textbf{\textit{knife}}} \\ Arg \quad = \left\{Arg1 \quad = x : tool\right\} \\ Qualia \quad = \left\{\begin{array}{l} Telic \quad = cut(e,x,y) \\ \ldots \end{array}\right\} \\ \ldots \end{array}\right\}$$

Here, in telic fields, the parameter e is of an event-type, the second parameter is an object and second is a subject of an action. When analysing examples of selectional coercion in Section 5.2, the telic qualia field will be our main focus for distinguishing e.g., *started reading a book* from *started a book*.

Agentive: According to Aristotle (Physics II), "coming into being", the way an object is created, is an essential mode of explanation of it. Knowing how objects are "brought to" or created is important to distinguish them for example in the cases where other quales coincide. In nouns the *Agent* field is usually an event predicate with noun's *Arg* as one of its arguments. For example, a cake is treated an artefact which is being created through the process of baking. On the other hand, baking process in relation to potatoes is just a change-of-state predicate and therefore it does not belong to an agentive quale of theirs:

$$\left\{\begin{array}{l}\underline{\textbf{\textit{cake}}} \\ Arg = \left\{Arg1 = x : food \quad\right\} \\ Qualia = \left\{\begin{array}{l} Agent = bake(e,y,x) \\ \ldots \end{array}\right\} \\ \ldots \end{array}\right\}$$

4.3 Generative Lexicon: Compositional Mechanisms

With the Qualia field, we can analyze inheritent meaning shifts of nouns. These shifts occur when the noun itself in a phrase refers not to the material object from its Arg field but it rather refers to one of the modes of explanation of the noun i.e. one of its lexical roles. Consider the phrase *"she started a book"*. Here the shift of meaning from just *"start a book"* to either *"start reading a book"* or *"start writing a book"* in the verb part is obvious as all books are closely tied with the process of reading (or writing) them. The fact that the predicate $read(x, book)$ forms the telic field of the record for "*book*" conveys exactly this idea. Then the word record of *"start"* combined with the word record of *"book"* will be able to pull out either telic or agentive field and make itself applied to one of them yielding a correct final interpretation of *"start(reading(book))"* or *"start(writing(book))"* instead of *"start(book)"* which is ill-typed (as you can "*start*" only something which is an "*event*").

Pustejovsky proposes several compositional rules which can alter and compose elements of the lexical structure of constituents in order to deal with initial typing mismatches, constraining well-formedness of phrases by using lexical (and not only semantic) information and provide more fine-grained analysis:

> A major consequence of this <i.e. compositional> approach is that the isomorphism between syntactic and semantic categories cannot be maintained for all levels of linguistic description, nor is it desirable . What this means is that a syntactic phrase cannot be interpreted outside of the syntactic and semantic context within which it appears. Rather, only by embedding the phrase can the appropriate denotation be determined. (Pustejovsky 1996 p.105)

In Pustejovsky 1996 (ch. 7) three main mechanisms are given: *coercion*, *co-composition* and *selective binding*.

Coercions (Pustejovsky 1996, 7.3) in the Generative Lexicon can be understood as evoking a similar idea to coercions in MTTs we discussed earlier in Section 3.1.2. Coercion in general is a shift of a constituent's semantic type. In MTTs they are defined as maps with certain proof-theoretic properties (such as uniqueness of the map). In the Generative Lexicon, apart from the use in subtyping, coercions are used in a direct way: for shifting a word's interpretation to another one which is known to be the *alias* of the initial direct meaning. The alias can be taken from the Qualia of the word e.g. in the phrase "*to start a book*" the word "*book*" is coerced to events of "*reading a book*" and "*writing a book*" which are both situated in the qualia and therefore aliased with the *Arg* head (which is the book itself) and suitable for the coercion. We see a selective nature of the meaning generation with coercions: if a verb is combined with an argument which semantic type does not suit the process of coercion can be started by searching lexical aliases of the argument which, after being coerced to, will fulfil verbs argument's typing.

Co-composition (Pustejovsky 1996, 7.2) describes one case of verbal polysemy as in phrases:

(41) John *baked* a potato.

(42) John *baked* a cake.

We can see that in (41) the verb "*bake*" denoted a process while in 42 it denotes a transition. This difference in meanings can be captured by the presence of the "*bake*" in the agentive field of cake's qualia: baking is essential for its come-to-being.

The definition which allows to formalize this phenomena without introducing type-shift for the verb is the following (Pustejovsky 1996 p.124):

> *Function application with qualia:*
> For two expressions α of type $a \to b$ and β of type a, with qualia structures QS_α and QS_β, respectively, if there is a quale value shared by α and β, $[_{QS_\alpha} \ldots [Q_i = \gamma] \ldots]$ and $[_{QS_\beta} \ldots [Q_i = \gamma] \ldots]$, then we can define the qualia unification of QS_α and QS_β, $QS_\alpha \sqcap QS_\beta$, as the unique greatest lower bound of these two qualia structures. Further, $\alpha(\beta)$ is of type b with $QS_{\alpha(\beta)} = QS_\alpha \sqcap QS_\beta$.

Thus, the structure of the result reflects aspects of both constituents highlighting special roles of the VP (verb phrase) head (i.e., *bake* in *bake a cake*) in the coordination with an

argument.

Selective binding (Pustejovsky 1996, 7.3) deals with adjectival polysemy. Consider the following examples:

(43) We will need a *fast* boat to get back in time.

(44) John is a *fast* typist.

(45) Fast drivers will be caught and fined.

In each of these three sentences the adjective conveys different meanings. (45) says about the potential speed of boat's movement, (44) says about speed of committing an action of typing the text and the phrase (45) denotes a habit of a person to drive on high speeds. We can see that adjectives like "*fast*" are not globally ambiguous but their polysemy strongly depends on the semantics of the NP head it modifies. As before that means that the specification of different typings and definitions of words which are polysemous in this contingent way is excessive and therefore the selective rule is proposed (Pustejovsky 1996 p. 129):

> *Selective binding:*
> If α is of type $a \to a$, β is of type b, and the qualia structure of β, QS_β has a quale q of type a, then $\alpha\beta$ is of type b, where $[\![\alpha\beta]\!] = \beta \cap \alpha(q)$

Now we can see that the object which is being modified by an adjective can treat the adjective as a function and apply it to one of the conforming qualia fields it has. The typing of the adjective stays the same and selecting mechanism finds a place for it to get properly applied generating desired meanings such as "$fast(typing)$", "$fast(driving)$" etc. instead of the general style of intersective-style treatment as in "$typist(x) \wedge fast(x)$" which, before all, does not carry as much of semantic preciseness.

5 Proposal: Lexical Conceptual Paradigm in MTT's

In this chapter we propose a way to emulate lexical records for nouns along the ideas of Pustejovsky just presented in Sections 4.2, 4.3 and discuss several approaches for modelling selectional coercion from these nouns to their lexical roles. We propose some prima facie plausible proposals for implementing this in MMTs, enriched with GL lexical semantic structures.

Also we discuss a possibility for the implementation of ambiguity analysis in which, for instance *begin a book* can be genuinely ambiguous between *begin (reading) a book* and *begin (writing) a book*. However, we show that, unfortunately, this is not compatible with the system both allowing ambiguity implementations and remaining coherent (see Definition 2 for the definition of coherence). Therefore, accounting for such ambiguities remains an challenge to be addressed within (and outside) MTT.

In Sections 5.1.2 and 5.1.1 we present the challenges outlined in Section 1.2 in more detail.

In Section 5.2, we discuss the use of Σ-types for the purpose of defining complex typings for nouns which would contain the lexical information in a way suitable for further compositions. In other words, we integrate the Lexical Conceptual Paradigm into MTT via Σ-types.

In 5.3, we pose two ways to use our newly defined lexical records of nouns in meaning derivations: via wrapper functions and via coercing to secondary projections of tuples and dependent unit types. However, we show that none of these is consistent with coherence if we want to accommodate ambiguity analysis.

5.1 The two challenges

Here we outline the two challenges we address in our proposal.

5.1.1 Lexical records and selectional coercions

The constructions we are going to investigate in the following section of this thesis are motivated by the integration of Qualia records from Generative Lexicon into Modern Type Theories. This challenge consists of two main parts:

(C1) The organization of new types assigned to nouns.

(C2) Developing compositional mechanisms which would interact with newly-defined noun-types.

A solution to (C1) would be a definition of a new type assigned to a noun interpretation which would preserve the original system's behaviour (e.g. CNs-as-Types principle) while effectively organizing word's meta-properties in a concise way along the representation given in Generative Lexicon. Records in GL are defined as labeled sets (as well as, for example, records from Type Theory with Records in e.g. R. Cooper 2023) and MTTs do not have an ability to use sets in a straightforward manner. Therefore one of the puzzles of finding a suitable solution is in the emulation of the storage of information inside of the typing signatures of noun witnesses. Moreover, not only the record we define should look descriptive but it should also be used for compositional meaning derivation. This leads to the second challenge (C2).

The development of compositional mechanisms for record types and selectional coercion, as stated above in (C2), is the second challenge and it is clearly interlined with (C1). After adopting the representation of lexical information we should also adopt in a sound manner the compositional mechanisms proposed for that records in Generative Lexicon (we described them in Section 4.3).

Thus, by having both types emulating qualia and an automatic means of composing those types which generate the desired meanings, we may be able to introduce an alternative way to model selectional coercions in MTTs. This can greatly contribute to the lexical transparency of linguistic coercion classification and modelling as now the distinction between contextual and, additionally, selectional coercions will be visible as the use of lexical information will be highlighted.

5.1.2 Ambiguous selectional coercions in MTT.

In this work we also consider a second challenge which follows from the first one. It concerns the fact that some selectional coercion constructions are ambiguous.

Consider the case of the phrase *"Mary starts a book"* where the selectional coercion occurs. In a standard MTT approach (e.g. Chatzikyriakidis and Luo 2020 5.2) the noun *"book"* is interpreted with the type $Book$ which is defined to be a subtype of $Phys \bullet$

Inf, a dot-type of a physical object and an informational object, and the verb *"start"* is interpreted with the following type:

$$[\![start]\!] : \Pi h : Human.(Evt_A(h) \to Prop)$$

As we can see, for the constant $Mary$ of type $Human$ denoting Mary, the domain of $start(Mary)$ is $Evt_A(h)$ and not $Phys \bullet Inf$. Therefore an expression $start(Mary, b)$, where b is of type $Book$, is ill-typed as it should not be the book itself that is getting "started" but the process of reading or writing it. In MTTs to parse the case of linguistic coercions such as this one, a parametrized coercion map $c(Mary) = reading(Mary) : Book \to Evt_A(Mary)$ is utilized (Asher and Luo 2013). The linguistic coercion is modelled in the fragment definition via the family of subtyping relation $Book \leq_{reading(h)} Evt_a(h)$ indexed by h of type $Human$. The use of the coercive subtyping results in the following equality being well-typed:

$$start(Mary, b) = start(Mary, reading(Mary, b)) : Prop.$$

We argue that in the real situation without any specified context the analysis of the phrase above allows both of the linguistic coercions:

(46) $[\![Mary\ starts\ a\ book]\!]_1 = start(m, writing(m, book))$

(47) $[\![Mary\ starts\ a\ book]\!]_2 = start(m, reading(m, book))$

In other words, we claim that in general, the meaning of this phrase is ambiguous. This differs from the analysis presented in Chatzikyriakidis and Luo 2020 (pp.59-60), in which *Mary started a book* is interpreted as *Mary started writing a book* if Mary is the author of the book and *Mary started reading a book*, otherwise. I.e., it is ruled out semantically that *Mary started a book* can mean *Mary started reading a book* if Mary wrote the book. Given that a plausible interpretation of *start a book* is that it is ambiguous between 'start reading' and 'start writing' (where the resolution of this ambiguity would be resolved pragmatically, not semantically), an interesting question to ask is whether such an ambiguity analysis can be implemented in MTT. The challenge this introduces, however, relates to coherence. Without the exact context we can not choose only one of the possible meanings and therefore both interpretations (46) and (47) should be allowed even for the phrases describing the book and its real author as in "Tolstoy started War and Peace" (maybe it is implied that he decided to re-read his own novel before publication).

However, it is impossible to allow such ambiguity by means of the coercive subtyping method, since the *coherence* property should be maintained. Coherence is crucial for coercive subtyping and it requires coercions between two types to be unique. Therefore, as shown in Chatzikyriakidis and Luo 2020 (pp.58-60), if for some agent m both $reading(m)$ and $writing(m)$ are defined as coercions then every object b of type $Book$ can be coerced to two different events of the same type $Evt_A(m)$. Hence both subtypings $Book \leq_{reading(m)} Evt_A(m)$ and $Book \leq_{writing(m)} Evt_A(m)$ cannot be present in the fragment at the same time as they do not satisfy the uniqueness property of a coercion. For that, the coercion c described in the Definition (40) of Section 4.1 is introduced. Depending on both the subject and object of the action and the contextual relation between them (namely, whether the subject wrote the book or not) the coercion c either becomes *reading* or *writing*. Parameterizing the coercion extends the expressivity as the appropriate coercion form can be chosen differently for each agent. However, this approach is not compatible with a genuine ambiguity, since it prevents e.g., 'Tolstoy started W&P'

from being interpreted as 'Tolstoy started reading W&P'. In summary, the challenge of attempting to provide an ambiguity-based analysis is how to implement this without violating coherence. We will review alternatives for addressing this challenge in the remains of this section. However, we will show that none of the prima facie plausible alternatives for doing so are viable. That said, even though a solution was not found, the proposed organization of lexical information of nouns in MTTs allows us to successfully analyze phrases which do not imply ambiguity, and so we are still able to account for some examples of selectional coercion and differentiate it as a phenomenon from contextual coercion.

5.2 Lexical Records in MTTs

In the following subsections we integrate the LCP into MTT.

In 5.2.1, we review the alignment of lexical information mentioned in Luo 2011a and finally we define our own proposal on lexical records in the section 5.2.2.

5.2.1 The Lexical Conceptual Paradigm (LCP) in MTT

In Modern Type Theories, Σ-types allow the building of tuples of dependent types, since Σ-types can be nested. For example, the type $\Sigma x_1 : A_1.\Sigma x_2 : A_2(x_1).\ldots\Sigma x_{n-1} : A_{n-1}(x_1, x_2, \ldots).A_n(x_1, x_2, \ldots)$ can be represented as in (48):

$$(48) \quad \left\{ \begin{array}{ll} x_1 & : A_1 \\ x_2 & : A_2(x_1) \\ \ldots & \\ x_n & : A_n(x_1, \ldots, x_{n-1}) \end{array} \right\}$$

By nesting Σ-types it may be possible to emulate *records* for each noun which would organize possible objects which are available for coercing to them (in the linguistic sense). Thus, the information about coercion will be stored "closer" to nouns and also stored without the direct use of a coercive subtyping mechanism. This might help us in the challenge of ambiguity analysis and more fine-grained lexical alignment of the system.

As a first step, we will integrate LCPs from the generative Lexicon into MTT (see Section 4.2 for a description of LCP). Doing so might contribute to ambiguity analysis as well as to help organize the system with the consideration of the lexical information of the nouns. In Luo 2011a (5.2) it is hinted that we can consider the typing of the LCP-entry of the noun "*book*" realized with the use of Σ-types[7]:

$$(49) \quad [\![book]\!] := \left\{ \begin{array}{ll} Arg & = x : Phys \bullet Inf \\ Qualia & = \left\{ \begin{array}{ll} Formal & = Hold(p_1(Arg), p_2(Arg)) \\ Telic & = R(Arg) \\ Agent & = \exists h : Human.W(h, Arg) \end{array} \right\} \end{array} \right\}$$

Here $R(x)$ stands for "x to be read" and $W(x, y)$ stands for "x wrote y" and p_1 and p_2 are the corresponding projections of the dot-type.

[7]Note that this construction lacks the $Phys \bullet Inf\_lcp$ which would have to be present in Pustejovsky's definition of LCP. As the argument is typed as $Phys \bullet Inf$ in MTT it already forces a behaviour we want due to the powerful type system of MTTs.

Records in the Generative Lexicon are represented by labeled sets of typed elements. In MTTs, due to the propositions-as-types principle, objects of type *Prop* can be stored inside Sigma types, in a certain sense emulating the structure of a set (as propositions are types and their objects can be considered as proofs of those propositions), formally resulting in (following the definition (49) proposed in Luo 2011a):

$$\begin{aligned} Hold &: Phys \to Inf \to Prop, \\ R &: Inf \to Prop, \\ W &: Human \to Prop, \end{aligned}$$

$$\begin{aligned} [\![book]\!] :&\Sigma arg : Book. \\ &\Sigma f : Hold(p_1(arg), p_2(arg)). \\ &\Sigma t : R(arg).\exists h : Human.W(h, Arg). \end{aligned}$$

Consider an object b of the complex type we just defined. Then the projection to the last entry of the nested Σ-typed element will be of type $\pi_2(\pi_2(\pi_2([\![book]\!]))) : \exists h : Human.W(h, Arg)$ i.e. the typing of the element already holds the information that the element is the proof of the fact that the book in question itself is written by an existing human.

Note that such complex definitions of nouns as nested tuples still allows the usual use of the noun-as-a-type as the first projection can be chosen as a coercion:

$$\Sigma arg : (Phys \bullet Inf).(\ldots) \leq_{\pi_1} arg : (Phys \bullet Inf).$$

While the proposed typing helps to carry lexical information inside the type system the compositional operations cannot be implemented in that exact manner as they are implemented for usual MTT typings because of all the qualia are being typed as *Prop*. There is a need for Σ-type secondary projections to be only of type *Prop* so as to not to lose any information, *these are objects of Σ-types which are tuples of objects, not the types themselves in the general case.* Every object (a, b) of some type $\Sigma x : A.B(x)$ is a pair of objects of types A and $B(x)$. If $B(x)$ is a type *Event* we will never be able to decide which exact object b of this type would be in the pair (a, b) and hence we would not be able to recover the exact description of the event just by the typing of the Σ-type to which it belongs. On the other hand, if the typing of the second tuple entry is $Reading(a)$ of the universe *Prop* then any element of this type would be a proof of the desired proposition due to the Curry-Howard principle.

However, in order to use qualia for meaning derivation along the pattern of the analysis already used in MTTs all the records need to be of type *Event* or one of its parameterized subtypes. To satisfy the typing (50) below of the verb "*start*", the second argument - which is the event that is being started - should be of type $Evt_A(h)$. Hence, just knowing that there is a proposition $R(b)$ associated with the book b (as in (49)) would not help us get the desired object of type $Evt_A(h)$: at first, the proposition does not carry any information about the agent (in this case the reader) and most importantly it is just the *Prop* and therefore cannot be generalized and combined with the verb.

(50) $[\![start]\!] : \Pi h : Human.(Evt_A(h \to Prop)$

In the Generative Lexicon, on the other hand, the telic quale for "read(b)" is defined as "$\lambda x.reading(x, b)$" and we can see that it is left underspecified - it requires further composition with the sentence subject, some human x, to be completed into the shape $reading(x, b)$ denoting "x is reading b" and thus carrying more semantic information. In combination with the neo-Davidsonian approach, the quale $\lambda x : Human.read(x, b)$ can be typed as $\lambda x.Evt_A(x)$. Then the resulting event can be used for subsequent combination with other modifiers but, as we stressed before, the information about the exact event is lost. A possible solution which manages to preserve the information is proposed in the next section.

5.2.2 Event Semantics for Qualia Records

Given the problems just mentioned, an avenue to pursue the lexical information organization is to examine whether we can use the neo-Davidsonian approach from Luo and Soloviev 2017 (briefly described in the Section 3.3.5 of the current work) combined with the nested Σ-type solution to emulate Generative Lexicon's Qualia records.

An event-semantic approach realized in a Montagovian setting uses elements of a distinct type *Event* which obtain a meaning in a Davidsonian way through predicates of form $Event \to \boldsymbol{t}$ such as $reading(x)$, $running(x)$ along with the adverbial modification which is also defined as $Event \to \boldsymbol{t}$, for example $quickly(x)$, $skilfully(x)$, $suddenly(x)$. Agents and patients of the action are marked through predicates of type $Event \to \boldsymbol{e} \to \boldsymbol{t}$ such as $agent(x, y)$ and $patient(x, y)$. In MTTs (e.g. Luo and Soloviev 2017), as we mentioned, agent and patient notations are incorporated into the type itself via the dependent type definition while the essence of events is defined with predicates of form $Event \to Prop$ or, with more alignment, as $\Pi x : T.(Evt_A(x) \to Prop)$, where T is some type of CN universe and $Evt_A(x)$ denotes and event whose **A**gent is x, for example, $bark : \Pi x : Dog.(Evt_A(x) \to Prop)$ ((13) in Luo and Soloviev 2017) defines an event of barking and it is constrained only for objects of the type *Dog*.

On the other hand, when modelling linguistic coercions from nouns to events (in Chatzikyriakidis and Luo 2020(pp. 58-60)) event-defining coercions are set to be of type $reading : \Pi.h : Human.(Book \to Evt_A(h))$ in a particular example considered in that section. There, $reading(h)$ is set to be a coercion $Book \leq_{reading(h)} Evt_A(h)$ which fulfils the typing of verbs like *enjoy* or *start* which are of shape $\Pi h.Human.(Evt_A(h) \to Prop)$. Therefore different events, like running, reading, barking, are treated as elements of corresponding event-types as $read(John) : Evt_A(John)$ and not through predicating over generally unspecified elements of type *Event*.

In order to use events as parts of the lexical record of a noun we define them as predicates of event-types into the *Prop* universe:

(51) $read : Event \to Prop$,

(52) $talk : Event \to Prop$,

(53) $run : Event \to Prop$.

or in a more precise manner (as in Luo and Soloviev 2017) as:

(54) $read : \Pi h : Human.(Evt_A(h) \to Prop)$,

(55) $talk : \Pi h : Human.(Evt_A(h) \to Prop)$,

(56) $run : \Pi a : Animal.(Evt_A(a) \to Prop)$.

Due to established subtyping relations (from Section 3.3.5) any object of one of the dependent types $Evt_{AP}(a,p)$, $Evt_A(a)$, $Evt_P(p)$ still fits with more general typings of predicates (51)-(53) and $Evt_{AP}(a,p)$ fits typings of (54)-(56).

As we can now give events their meanings through propositional predicates and therefore employ the CNs-as-types principle we are able to store event information on a typing level of $\Sigma-$entries which emulate lexical records from the Generative Lexicon. We give two examples, *skillful typist* and *start a book* below to show how this proposal works. We will only describe the structure of the lexical records for nouns and show that desired analyses of the phrases are well typed. Later, in Sections 5.3.2 5.3.3 we discuss the compositional mechanisms which can be introduced to generate those analyses.

Example: *skillful typist*

We consider the noun "*typist*" as one possessing a single quale representing the process of typing that each typist conducts. Therefore we model the noun's lexical record by defining the interpretation of the noun with the type $TypistQ$ (the letter Q stands for Qualia):

$$[\![typist]\!] = TypistQ = \Sigma x : Typist.\Sigma e : Evt_A(x).typing(x,e) : Type$$

where:

$$typing : \Pi x : Agent.(Evt_A(x) \to Prop)$$

With the given definition every (canonical) object t of type $TypistQ$ is a triple (x,e,p) where x is the entity of the typist (i.e. Arg field of the record), e is some event conducted by x and p is a *proof that event e is exactly the typing process of the typist x*. The lack of the exact semantic information of the event record is compensated by always keeping the event together with the proposition which describes it (as in the neo-Davidsonian approach). As propositions are types, the description of the event is getting fixed on the typing level already.

If we analyze a process of meaning-derivation happening with an adjective "*skilful*" which is allowed to address the qualia of a noun it modifies we can consider an expression (57) which may be formed along the "selective binding" rule idea from Pustejovsky 1996 and interpret an expression "skilful typist":

(57) $[\![\text{skilful typist}]\!] = \Sigma t : (\Sigma x : Typist.\Sigma e : Evt_A(x).typing(e)).skilful(\pi_1(\pi_2(t)) : Type,$
i.e. $[\![\text{skilful typist}]\!] = \Sigma t : TypistQ.skilful(\pi_1(\pi_2(t)).$

The expression (57) is well-typed considering $skilful$ is actually an event modifier of type $Event \to Prop$, not the subsective adjective as it was before in MTTs (along the lines of the Section 3.3 of Chatzikyriakidis and Luo 2020):

$$\dfrac{skilful : Event \to Prop \qquad \dfrac{t : \Sigma x : Typist.\Sigma e : Evt_A(x).typing(x,e) \quad \ldots}{\pi_1(\pi_2(t)) : Evt_A(x)} \qquad Evt_A \leq Event}{skilful(\pi_1(\pi_2(t))) : Prop}$$

Hence, the witness of the type $\Sigma t : (\Sigma x : Typist.e : Evt_A(x).typing(x,e).skilful(\pi_1(\pi_2(t)))$ would be a tuple $\langle\langle x, e, p_2(e)\rangle, p_2(e)\rangle$ where x is a human who is a typist (of type $Typist$),

e is an event conducted by x (this relation is fixed by the typing $Evt_A(x)$), $p_1(t, e)$ is a proof of the fact that "e is typing" and p_2 is a proof of the fact that "e is skilful".

A more constrained version can be set by means of Π-polymorphism and dependent event-types. We may set the predicate $skilful$ to only be applied to events conducted by humans via the following typing:

(58) $skilful : \Pi.h : Human.(Evt_A(h) \to Prop)$

then, by contravariance and a subtyping relation $Typist \leq Human$ the interpretation (58) can be regarded as:

(59) $skilful : \Pi.h : Typist.(Evt_A(h) \to Prop)$

and the expression $skilful(t, \pi_1(\pi_2(t)))$ would still be well typed as we treat $skilful$ as in (59) and also coerce an object t of the new compound type to $x : Typist$ through the first projection π_1 (see the proofs in Figure (3) with contexts, signatures and entailment relation omitted for compactness). We have shown that under certain assumptions, 'skillful typist' is well-typed. However, as we will see in Sections 5.3.2 and 5.3.3, other puzzles arise when we look more closely at the semantics of 'skillful' and try to get the expression $skilful(t, \pi_1(\pi_2(t)))$ from the initial $\exists t : TypistQ.skilful(t)$ automatically.

Definitions:

$[\![skilful]\!] = skilful : \Pi h : Human.(Evt_A(h) \to Prop)$
$[\![typing]\!] = typing : \Pi h : Human.(Evt_A(h) \to Prop)$
$[\![typist]\!] = TypistQ = \Sigma x : Typist.\Sigma e : Evt_A(x).typing(x, e)$

$$(\mathbf{D})\quad \frac{skilful : \Pi.x : Typist.(Evt_A(x) \to Prop) \quad \Sigma x : Typist.(\ldots) \leq_{\pi_1} Typist \quad \dfrac{t : \Sigma x : Typist.\Sigma e : Evt_A(x).(\ldots)}{\pi_1(\pi_2(t)) : Evt_A(\pi_1(t))}}{\dfrac{skilful(t) : Evt_A(\pi_1(t)) \to Prop}{skilful(t, \pi_1(\pi_2(t))) : Prop}}$$

$$\frac{\ldots \quad \dfrac{typing : \Pi h : Human.(\ldots) \quad \ldots \quad x : Typist \quad Typist \leq Human}{typing(x, e) : Prop}}{\dfrac{\Sigma x : Typist.\Sigma e : Evt_A(x).typing(x, e) : Type}{\mathbf{\Sigma t} : (\mathbf{\Sigma x} : \mathbf{Typist}.\mathbf{\Sigma e} : \mathbf{Evt_A(x)}.\mathbf{typing(x, e)}).\mathbf{skilful}(\pi_{\mathbf{1}}(\pi_{\mathbf{2}}(\mathbf{t}))) : \mathbf{Type}}} \quad \frac{\dfrac{\mathbf{D}}{skilful(\pi_1(\pi_2(t))) : Prop}}{skilful(\pi_1(\pi_2(t))) : Type}$$

Figure 3: well-typedness of "skilful typist"

Example: *start a book*

As there is a selective coercion occurring in the analysis of the phrase "*to start a book*", a definition for the noun "*book*" which would have qualia information incorporated can be beneficial as well. The difference from the previous example of the noun "*typist*" is that here the noun "*book*" has two qualia and both of them require an agent to be passed, as we discussed earlier in this work. However, a new complex record $BookQ$ which interprets the noun can be defined in a similar way to the noun "*typist*":

$$\begin{aligned}[\![book]\!] = BookQ = \Sigma x : Book.\Sigma e_1 : (\Pi h : Human.Evt_{AP}(h, x)).\\ \Sigma e_2 : (\Pi h : Human.Evt_{AP}(h, x)).\\ \Sigma p : (\forall z : Human.reading(z, x, e_1(z)).\\ (\forall z : Human.writing(z, x, e_2(z)))\end{aligned} \tag{60}$$

where:

$$reading : \Pi h : Human.\Pi x : Inf.(Evt_{AP}(h, x) \to Prop)$$
$$writing : \Pi h : Human.\Pi x : Inf.(Evt_{AP}(h, x) \to Prop)$$

As we can see, (underparametrized) events e_1, e_2 are kept in the record together with the propositions fixing their meaning, namely, that e_1, after an object of type $Human$ is applied to it, is a reading event and e_2 is a writing event.

Informally any object b of type $BookQ$ can be expressed as a record similar to what we've seen before with additional propositions fixing the nature of underspecified events in the qualia.

$$b = \left\{ \begin{array}{ll} Arg & = x : Book \\ Qualia & = \left\{ \begin{array}{ll} Telic & = e_1 : \Pi h : Human.Evt_{AP}(h, Arg) \\ & \text{s.t. } \forall z : Human.reading(z, Arg, Telic(z)) \\ Agentive & = e_2 : \Pi h : Human.Evt_{AP}(h, Arg) \\ & \text{s.t. } \forall z : Human.wrting(z, Arg, Agentive(z)) \end{array} \right\} \end{array} \right\}$$

The following definition also emulates the original Generative Lexicon's structure of the qualia records of nouns as there the telic and agentive qualia fields contain underspecified events: for instance an interpretation of the noun "novel", as we mentioned before, is given in Pustejovsky 1996(p.79) as follows:

$$\lambda x[novel(x) \ldots \text{TELIC} = \lambda y.[read(y, x)] \ldots].$$

Hence now by having $reading(b) : \Pi h : Human.Evt_{AP}(h, b)$ as the element to which the quale unit coerces we successfully formed a "record" for the noun very close to the proposal from Pustejovsky 1996.[8]

[8]Another proposal might be to organize noun records which have more than one quale in the shape $\Sigma x : A.\Sigma q_1 : (\Sigma e : Evt.Prop).(\Sigma e : Evt.Prop)$. In this way each quale would be interpreted as a pair of event and a proposition as opposed to the definition we use in this work when at first all the qualia events go followed by the "tail" of propositions defining them. With setting first projections as coercions this alternative definition can be used exactly in the same way.

Then, if we follow the Generative Lexicon's compositional approach we can get interpretations (61) or (62), where the type $BookQ$ denotes a new record-like type for the noun "*book*" from (60). The projection $\pi_1(\pi_2(b))$ in the expression (61) computes to the event representing the reading of the book and projection $\pi_1(\pi_2(\pi_2(b)))$ in (62) computes to the event representing the reading of the book.

(61) $\exists b : BookQ.start(Mary, \pi_1(\pi_2(b))(Mary)) : Prop$

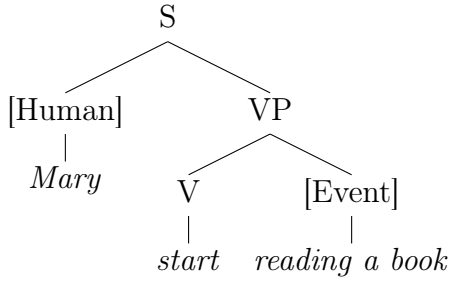
(62) $\exists b : BookQ.start(Mary, \pi_1(\pi_2(\pi_2(b)))(Mary)) : Prop$

with the verb "*start*" being interpreted as before as:

$$[\![start]\!] := start : \Pi h : Human.(Evt_A(h) \to Prop).$$

And the witness of the proposition is a proof of the fact that there is an object $b = \langle x, e_1, e_2, p_1, p_2 \rangle$ which represents a book, two of the associated events and their descriptions p_1 and p_2 along with the proof of $start(Mary, e_1(Mary))$ or $start(Mary, e_2(Mary))$. Having two objects, the tuple representing the book and the proof of the verbal expression, we obtain knowledge about the situation expressed in the phrase, that is, a situation where there is an object x of type $Book$ and the event $e_1(Mary)$ (or $e_2(Mary)$) of reading (or writing) the book x by Mary is being started. Note that typings of $e_1(Mary)$ and $e_2(Mary)$ carry the information about the fact that Mary is an agent and the book x is a patient.

Following the syntactic structure of the phrase "*Mary starts a book*" we may assume that "starts" as a predicate has two arguments, first is the subject and the second one is an event (to which the word book is getting linguistically coerced to). The second argument, event, is formed in the syntactic tree independently from the first one and therefore it may represent only an event in which a particular book is being read, but without the information of the agent who reads it:



That being said, we might use a different typing for the interpretation of the verb "start", namely one which does not ask for an event conducted by the first argument, but instead for an underspecified event of type $\Pi h : Human.Evt_A(h)$ as in:

$$start : Human \to (\Pi h : Human.Evt_A(h)) \to Prop$$

Such typing still lets us disallow incorrect phrases such as "*to start a typist*" or "*skilful book*" from being well-typed because now qualia of these nouns wouldn't match the corresponding predicate typings: one is underspecified and requires a subject to be passed and the other one is already a (sub)type of $Event$.

Then in order to interpret the coercion of the noun "*book*" to the event in the context of *start* we just need to project it to one of the qualia it has, without applying a subject

to the projections as we did before in 61) and (62):

$$[\![\text{Mary starts a book}]\!] = \exists b : BookQ.start(Mary, \pi_1(\pi_2(b)))$$
$$= \exists b : \left\{ \begin{array}{ll} x : & Book \\ e_1 : & \Pi h : Human.Evt_{AP}(h, x) \\ p_1 : & \forall z : Human.reading(z, x, e_1(z)) \\ & \ldots \end{array} \right\}.start(Mary, e_1)$$

A detailed description of this example is displayed in Figure 4.

Definitions:

$$
\begin{aligned}
&[\![Mary]\!] = Mary : Human \\
&[\![start]\!] = start : Human \to (\Pi h : Human.Evt_A(h)) \to Prop \\
&[\![reading]\!] = reading : \Pi h : Human.\Pi y : Inf.(Evt_{AP}(h, y) \to Prop) \\
&BookQ = \Sigma x : Book.\Sigma e_1 : (\Pi h : Human.Evt_{AP}(h, x)). \\
&\qquad\qquad\qquad \Sigma e_2 : (\Pi h : Human.Evt_{AP}(h, x)). \\
&\qquad\qquad\qquad \Sigma p : (\forall z : Human.reading(z, x, e_1(z))). \\
&\qquad\qquad\qquad\quad (\forall z : Human.writing(z, x, e_2(z))) : Type \\
&[\![the\ book]\!] = b : BookQ
\end{aligned}
$$

Proof:

$$
\begin{aligned}
&\varphi_1 = start(Mary) : (\Pi h : Human.Evt_A(h)) \to Prop \\
&\varphi_2 = \pi_1(\pi_2(b)) : \Pi h : Human.Evt_{AP}(h, \pi_1(b)) && \text{(from the definition of type } B) \\
&\varphi_3 = \Pi h : Human.Evt_{AP}(h, \pi_1(b)) \leq \Pi h : Human.Evt_A(h) && \text{(from the definition of dependent event type subtyping)}
\end{aligned}
$$

$$
\frac{\Gamma \vdash_\Delta \varphi_1 \quad \Gamma \vdash_\Delta \varphi_2 \quad \Gamma \vdash_\Delta \varphi_3}{\Gamma \vdash_\Delta start(Mary, \pi_1(\pi_2(b))) : Prop}\ CA_1
$$

Figure 4: well-typedness of “Mary starts the book”

5.2.3 Interim Summary

In this section we have outlined a means of integrating LCP into MMT, while still respecting the idea that modifiers like *skillful* and verbs like *start* take events as arguments. In the next section, we assess whether this proposal can be used together with some compositional rules of the type system in order to generate coerced meanings automatically. Also we investigate if we can capture the ambiguity of selectional coercions such as *start a book* mentioned in Section 5.1.2.

5.3 Formal Semantics with Lexical Records

In this section we propose several ways of working with lexical records realized through Σ-types in previous sections. We will see that the ambiguous readings of constructions like *start a book* is, unfortunately, not realizable solely through the already present inference rules of the system (Section 5.3.1). However, we also had a second aim, which was to account for selectional coercions in a way that demarcates them from contextual coercions. We show that if we set ambiguity aside we can find some solutions which would make the proposed definition work, even though for the restricted language fragment with nouns having only one quale or qualia of distinct types.

In 5.3.2 we show how new "wrapper" definitions of noun modifiers can confront the problem and in 5.3.3 we discuss possibilities of defining secondary projections (i.e. π_2's) to be coercions through certain workarounds such as parametrized coercions and dependent unit types.

5.3.1 (The lack of a) Pure Solution

In Section 5.2.2 we proposed a new typing for noun interpretations with a Σ-type tuple consisting of a noun object (of the unique noun type as it was before in MTTs) and its modes of explanation represented by event types together with propositions defining these events. We described the desired use of new lexically rich typings and proved that *if* the proper composition is somehow obtained it will be well-typed. The remaining challenge is the defining compositional mechanisms which can allow linguistic coercions from a noun to its qualia fields and hence forming desired judgments.

As outlined in Pustejovsky 1996 (Chapter 7) in the case of an initial mistyping of an expression the Generative Lexicon system may look for the suitable qualia field of an argument and coerce the argument to it (semantic selection) or just modify the qualia leaving the rest of the argument structure intact (selective binding).

According to the *semantic selection* phenomena discussed in Pustejovsky 1996 ch. 7.4 verbs like "*enjoy*" due to their type polysemy actually modify the qualia rather than an argument of a noun they apply to. It comes from the unnaturality of the verb having several different typings in order to get parsed in Phrases (63) and (64).

(63) John enjoys reading a book, $[\![enjoy]\!] : NP \to Event \to Prop$

(64) John enjoys a book, $[\![enjoy]\!] : NP \to NP \to Prop$

Thus, what might be happening here is the verb choosing the appropriate qualia of the type *Event* if an argument is ill-typed (i.e. if its a noun phrase and not an event). Then the result of the selection would be a transition from $enjoy(book)$ to $enjoy(reading(book))$

with the use of the coercion to the qualia field as illustrated in the following:

$$enjoy(book) \overset{\text{selective coercion}}{\longmapsto} enjoy(reading(book)).$$

Unfortunately, given an implementation of this process solely by means of MTTs definitions and coercive subtyping (as this is the only rule which allows to identify an object with its coercion image) would violate coherence in case of an underspecified context as it is allowed to get both $reading(book)$ and $writing(book)$ coercions by the nature of the Generative Lexicon idiom.

(65) $\Gamma \vdash_\Delta enjoy(book) = enjoy(reading(book)) : Prop$

(66) $\Gamma \vdash_\Delta enjoy(book) = enjoy(writing(book)) : Prop$

(67) $\Gamma \vdash_\Delta enjoy(reading(book)) = enjoy(writing(book)) : Prop$

Having (65) and (66) derivable with use of the hypothetical qualia coercion realization we will get (67) derivable through transitivity of equality. The well-formedness of (67) ruins the coherence of the subtyping relations: as $reading(book)$ is the proof object of the reading process and it surely differs from the proof object $writing(book)$ of writing it. So although we need to be able to differentiate proofs of $enjoy(reading(book))$ and $enjoy(writing(book))$, the initial $enjoy(book)$ is assumed to be coercable to both of them, and so they cannot be differentiated! If those objects are not violating coherence then they will indeed be equal as objects and then the linguistic meaning is absurd as (67) says that whenever a process of starting reading a book is the case it is also a case of starting writing the same book and, moreover, that a book reading event implies an event of writing it and vice versa.

One way to employ selective mechanisms and make use of the record-style noun definition might be possible within the restricted case where the hypothetical ambiguity is restricted. We will propose some approaches utilizing wrapper functions and dependent unit types in Sections 5.3.2 and 5.3.3.

If the ability to (linguistically) coerce to several different meanings of the phrase is valuable (e.g., if we want to allow *start a book* to be ambiguous between *start reading a book* and *start writing a book*), a more abstract solution is required to implement it in MTTs. Namely, some rule of formation of an expression which comes before the process of checking its derivability within the inference system is needed. We discuss this in the Section 6.

5.3.2 Wrapper Functions

If making space for the ambiguity of interpretations is not the goal, then in order to be able to make use of definitions with lexical records we can try to consider a restricted case of all nouns bearing no more than one qualia entry of each type (chosen by the context, for example) as otherwise the incoherence/absurdity issue described in the previous subsection arises. For instance, we can let the record for the noun "*book*" have either a $Telic$ or $Agent$ field, but not both, since both of them are of type $\Pi h : Human \to Evt_{AP}(h, Arg))$ which gives rise to the aforementioned coherence problems.

Constructions that we are defining in the following sections would still be useful for MTTs as they show how lexical information can be stored and used for the meaning derivation even though lacking the complete generality of the solution (i.e. lacking the ambiguity of certain selectional coercions).

The rule of selective binding, which applies the modifier to the qualia entry in case of the initial type mismatch is used to deal with some subsective adjectives such as "*fast*" or "*skilful*". These adjectives are indeed subsective but what they do is they modify some property of the subject instead of the subject itself via sense coercion and their nature is polysemous as properties can be different of sorts of events. This is unlike the subsective adjective "*small*" which modifies the size property of the subject instead of the subject itself. For *small*, no coercion is needed as it modifies only the size and nothing else: this connotation is already embedded in the adjective's meaning. Thereby, a desired analysis of "*skilful typist*", where a typist t is typed as $TypistQ$, a Σ-type of the entity and it's process of typing, should be of shape:

$$TypistQ := \Sigma x : Typist.\Sigma e : Evt_A(x).typing(x, e)$$
$$skilful(t) = \Sigma t : TypistQ.skilful(typing\_of(t))$$

One way to realize a compositional process that would allow this interpretation on the input "*skilful typist*" is to use a wrapper around the noun predicate. Instead of having an adjective interpretation typed only as $skilful : \Pi A : CN.(A \to Prop)$ or $skilful : (\Pi x : Agent.Evt_A(x)) \to Prop$ [9] we can overload it with two definitions where the first one is an event predicate and the second one references the first one:

$$[\![skilful]\!]_1 : Event \to Prop$$
$$[\![skilful]\!]_2 : (\Sigma x : Agent.Evt_A(x)) \to Prop$$
with the exact definition:
$$[\![skilful]\!]_2 := \lambda y : (\Sigma x : Agent.Evt_A(x)).[\![skilful]\!]_1(\pi_2(x))$$

Then, by using type overloading with unit types (Section 3.3.4) we can interpret the adjective as the sole member $skilful$ of the following type:

$$[\![skilful]\!] := skilful : \mathbf{1}_{skilful}$$

and then set coercions (68), (69) to both of the realizations $skilful_1$ and $skilful_2$.

(68) $\mathbf{1}_{skilful} \leq_{c_1} Event \to Prop$
$c_1(skilful) := [\![skilful]\!]_1$

(69) $\mathbf{1}_{skilful} \leq_{c_2} \Sigma x : Agent.Evt_A(x) \to Prop$
$c_2(skilful) := [\![skilful]\!]_2$

[9] Note that in this section we sometimes omit the "tails" of props at the end of noun record definitions as we can always use the fully-specified record (i.e. a Σ-type tuple of object, some events and some propositions about them) in the context requiring a Σ-type pair of an object and an event via the derivable subtyping below:

$$\frac{\vdash A \leq A \quad x : A \vdash \Sigma e : Evt_A(x).Prop \leq_{\pi_1} Evt_A(x)}{\vdash \Sigma x : A.\Sigma e : Evt_A(x).Prop \leq_c \Sigma x : A.Evt_A(x) : Type}$$

This subtyping can be easily generalized to the following one:

$$\Sigma x : A.\Sigma e_1 : E_1 \ldots \Sigma e_k : E_k.\Sigma p_1 : Prop. \ldots . Prop \leq \Sigma x : A.\Sigma e_1 : E_1 \ldots E_k$$

Now we can see that the expression $\Sigma x : TypistQ.skilful(x)$ will be well-typed as the realization $skilful_2$ takes a noun with a qualia attached and already set to be an event indexed by the same noun and then unwraps the pair and applies $skilful_1$ to the event:

$$\begin{aligned}[\![skilful]\!](x) &= skilful_2(x) \\ &= skilful_1(\pi_1(\pi_2(x)))\end{aligned}$$

The latter expression is well typed as previously shown in the Figure (3).

Combined with the MTT's mechanism of the modified adjectives formation we get a well-formed expression:

$$\Sigma x : (\Sigma t : Typist.e : Evt_A(t).typing(x, e)).skilful(e).$$

The typing of $[\![skilful]\!]_1$ requires an argument to be just of type *Event* and therefore acts as a quite general event modifier. Its down-point is that any event e can be combined with it into expression $skilful(e)$ possibly introducing overgeneration i.e., that events that cannot be called *skilful* would be admissible. But on the other hand, we have the ability to constrain the selectional coercion as we did in the definition of $[\![skilful]\!]_2$: setting an argument to be of type $\Sigma x : Agent.Evt_A(x)$, we impose a requirement on the thematic role of the coerced noun in the event, namely, we ask it to be an agent of the event selected for adjective modification. Thus, we automatically disallow interpretation of, for example, "skilful book" as all its qualia have the book entity acting in the thematic role of the patient.

The case of a meaning shift forced by verbs like "*enjoy*", "*like*", "*start*" can be treated in the similar manner considering nouns having only one quale. The typing of the verb and auxilary definitions should be the following:

(70) $[\![enjoy]\!] := enjoy : \mathbf{1}_{enjoy}$

(71) $[\![enjoy]\!]_1 : Agent \to \Pi x : Agent.(Evt_A(x)) \to Prop$

(72) $[\![enjoy]\!]_2 : Agent \to \Sigma x : CN.(\Sigma e : (\Pi a : Agent.Evt_A(a)).Prop) \to Prop$
with the exact definition:
$[\![enjoy]\!]_2(x, y) := [\![enjoy]\!]_1(x, \pi_1(\pi_2(y)(x)))$and with subtying relations:

(73) $\mathbf{1}_{enjoy} \leq_{c_1} Agent \to \Pi x : Agent.(Evt_A(x)) \to Prop$
$c_1(enjoy) = [\![enjoy]\!]_1$

(74) $\mathbf{1}_{enjoy} \leq_{c_2} Agent \to \Sigma x : CN.(\Sigma e : (\Pi a : Agent.Evt_A(a)).Prop) \to Prop$
$c_2(enjoy) = [\![enjoy]\!]_2$

The system will be able to choose a typing of the word upon encountering the expression $start(Mary, book)$. As the complex typing of the noun is compatible only with $[\![enjoy]\!]_2$ this definition would be selected via a call of the coercion c_2. Then during the computation of $[\![enjoy]\!]_2(Mary, book)$ the desired event-projection would be unfolded and applied to the usual definition $[\![enjoy]\!]_2$. The proof of the computed expression would be the same as shown in the Figure (4) and hence the resulting well-formed expression is (75).

(75) $$\exists b : \left\{ \begin{array}{ll} x : & Book \\ e : & \Pi h : Human.Evt_{AP}(h) \\ p : & \forall z : Human.reading(z, x, e(z)) \end{array} \right\}.start(Mary, \pi_1(\pi_2(b)))$$

This approach allows us to enrich the interpretation with a more fine-grained description as now information about aliased events is stored and displayed in the noun definition, either in the quantification argument (in cases of "some book" etc.) or in the declaration of a constant (when proper names as "Ulysses" are used for definite articles as in "the book"). Therefore event information is getting carried together with the noun allowing finely parametrized predicates to address it and, on the other hand, it does not affect using nouns only under their CN-universe typing due to the π_1-coercion.

However, while the adjective "*skulful*" which seems to strongly and essentially relate to some lexical meta-property of the noun it modifies rather than the noun itself, the adjective "$fast$" is a modifier of some moving ability of the noun object, but that ability to move is not always essential to the noun as a mode of explanation of it: "*fast driver*" really implies a person who drives fast and not the person who walks fast on their own while "*fast fox*" undoubtedly denotes an animal who moves fast but moving is not the qualia of a fox i.e. not its lexical meta-property. In this case adjective "fast" acts just as a general modifier of the moving speed and should be typed in the usual way as $\Pi A : CN.(A \to Prop)$. Therefore, adoption of an approach described before to analyze an adjective "*fast*" should force us to define three typings of the noun with the following general shapes::

(76) $[\![fast]\!]_1 : \Pi A : CN.A \to Prop$,

(77) $[\![fast]\!]_2 : Event \to Prop$,

(78) $[\![fast]\!]_3 : \Sigma x : Agent.Evt_A(x) \to Prop$.

Notice that the first interpretation (76) should be combined with the composition principle for subsective adjectives, i.e. the modified noun "*fast typist*" should be translated into MTT type $\Sigma x : Typist.[\![fast]\!]_1(Typist, x)$. Unlike (76) the other interpretations (77) and (78) should be used together with the compositional principle for intersective adjectives as we discussed before a modification of a qualia does not have to be tied to the noun Arg type - a human who types fast is still an animal who types fast, and a fast cyclist is still the animal who rides a bicycle fast et cetera. Therefore the result of the interpretation "*fast typist*" with the complex type $TypistQ$ needs to be formed as $\Sigma x : TypistQ.[\![fast]\!]_3(x)$ i.e. without the Π-polymorphism which locks the adjective. That said, the polymorphism of the definition of adjectives such as "*fast*" cannot be realized only through unit-type coercions because the application depends on the formation rules which are getting applied on the judgment formation level.

On the other hand, if we redefine $[\![fast]\!]_2$ and $[\![fast]\!]_3$ in the subsective manner, namely as $[\![fast]\!]'_2$ and $[\![fast]\!]'_3$, given in (79) and (80), we will be able to treat the initial composition in the same way as the composition of $[\![fast]\!]_1$ which is desirable. But, unfortunately, if we do so, we once again get problems with (in)coherence.

(79) $[\![fast]\!]'_2 : \Pi.A : CN.(Event \to Prop)$,

(80) $[\![fast]\!]'_3 : \Pi.A : CN.(\Sigma x : A.Event \to Prop)$.

Note that, at first, we have to sacrifice dependency in event types in (79), (80 as we have to parametrize adjectives by some type A from CN and we are not sure if it would be of type $Agent$ or not (and therefore not sure if $Evt_A(x)$ would be well-formed for $x : A$). And most importantly, the coherence would fail: consider coercions 81 and 82. Coercions c_1 and c_3 are fixed by us as they point the only element $fast$ of type $\mathbf{1}_{fast}$ to $[\![fast]\!]_1$ and

$[\![fast]\!]'_3$ respectively. Coercion c is present due to the subtyping properties of Π-types and the fact that the first projection of Σ-tuples is set to be a coercion.

(81) $\mathbf{1}_{fast} \leq_{c_1} \Pi A : CN.(A \to Prop) \leq_c \Pi.A : CN.(\Sigma x : A.Event \to Prop)$

(82) $\mathbf{1}_{fast} \leq_{c'_3} \Pi.A : CN.(\Sigma x : A.Event \to Prop)$

As we see from these typings maps $c \circ c_1$ and c'_3 should coincide to preserve coherence of the system but it is impossible for them to coincide.

In conclusion we can say that the pure method of using wrapper functions is applicable to some very small selection of constrained cases and shows a consistent way to use a proposed way of lexical information storage in MTTs but unfortunately it is not generalizable to any broader scope.

5.3.3 Projections as Coercions and Dependent Unit Types

There exists another pure approach of addressing qualia of nouns from a restricted fragment with only these records whose corresponding lexical records do not contain different objects of the same type (to avoid coherence violation). In such typings we may define qualia projections as coercions. This way, the object of a noun can naturally be coerced to the desired event quale on its own. Here, the linguistic (and type) coercion would be evoked by the initial type clash with the context and unlike coercing a noun entity (of its CN-as-types type) to the event the coercion of a record to its constituent event is happening. The difference between the two approaches is that in the first one we regard an objective witness of the noun-type (i.e. it's Arg member) as an event ("the physical witness of a book is also an event of reading it") and in the second one we regard an argument together with its modes of explanation as one of those modes ("the object which is embodied as a book and an event of reading it is also an event of reading it"). The method of declaring secondary Σ-projections π_2, $\pi_1 \circ \pi_2$ etc. as coercions still requires an implementation of several workarounds due to the issues promptly described in this section.

Parametrization of Σ-projections

Consider the case when the member of a Σ-record we want to coerce to is typed as $Event$. The case of coercing to this object is straightforward as the noun in question is just being associated with the event independent of any thematic roles and hence can be used in the corresponding contexts freely via coercion $\Sigma x : A.Event \leq_{\pi_2} Event$. The coercion declaration is possible because for the type $\Sigma x : A.Event$ the projection π_2 is well-typed as $\pi_2 : \Sigma x : A.Event \to Event$ without any complications as there is no dependency of $Event$ on A.

Speaking formally, if we regard Σ-types as ones defined through inference rules, projections π_1 and π_2 are not objects of a dependent type (i.e. a map) but mere notations of the type elimination and therefore by declaring a coercion to projections we actually imply the (homonymous) abbreviation of the wrapper map $\pi_i = \lambda x : (\Sigma x : A.Event).\pi_i(x)$. On the other hand, if we regard Σ-types as constants of the system LF_Δ we will truly have π_1 and π_2 defined as the following dependent maps(Chatzikyriakidis and Luo 2020, A5.2):

$\pi_1 = (A : Type)(B : (A)Type)(p : \Sigma x : A.B)A$

$\pi_2 = (A : Type)(B : (A)Type)(p : \Sigma x : A.B)[\pi_1(p)/x]B$

with exact computations $\pi_1(A, B(a,b)) = a$ and $\pi_2(A, B(a,b))$=b.

However, coercing to dependent event types (i.e $Evt_{AP}(x,y)$, $Evt_A(x)$ and $Evt_P(y)$) is problematic as it requires the coercion to be parameterized by the object from the first projection. To see this, consider a type $\Sigma x : A.Evt_A(x)$. The projection π_2 itself cannot establish a subtyping relation as it maps between objects $p : \Sigma x : A.Evt_A(x)$ and objects $e : Evt_A(\pi_1(p))$ i.e. the typing of its image depends on the input object. Hence, to make the direct coercion to projections work we might define the subtyping as follows with help of the following parametrization:

$$\Sigma x : A.Evt_A(x) \leq_{c(p)} Evt_A(\pi_1(p)),$$

where the coercion $c(p)$ is defined for objects $p : \Sigma x : A.Evt_A(x)$ as

$$c(p) := \lambda p' : (\Sigma x : A.Evt_A(\pi_1(p))).\pi_2(p') \tag{83}$$

We can see from these definitions that for any object p of the type $\Sigma x : A.Evt_A(x)$ the coercion $c(p)$ will be well defined and the full application would result in $c(p,p) = \pi_2(p)$.

Furthermore, the defined families of coercions between types $\Sigma x : A.Evt_A(x)$ and $Evt_A(a)$ for each $a : A$ are coherent as shown in the following proposition.

Proposition 1. *Consider an arbitrary object $p : \Sigma x : A.Evt_A(x)$. Then for any $p' : \Sigma x : A.Evt_A(x)$, if the object $c(p', p)$ is well-defined, it is equal to $c(p, p)$.*

Proof. Let q be of type $\Sigma x : A.Evt_A(x)$. Then, by canonicity we can treat p and q as pairs (a, e) and (a', e') respectively where $a, a' : A$ and $e : Evt_A(a)$, $e' : Evt_A(a')$. Clearly $c(p,p)$ is well defined and results in $c(p,p) = \pi_2(p)$ as:

$$\begin{aligned} c(p,p) &= c((a,e),(a,e)) \\ &= (\lambda p' : (\Sigma x : A.Evt_A(\pi_1(a,e))).\pi_2(p')))(a,e) \\ &= (\lambda p' : (\Sigma x : A.Evt_A(a)).\pi_2(p'))(a,e) \\ &= \pi_2(p) \end{aligned}$$

Now consider the map $c(q)$ or $c(a', e')$. If $p = q$ the proof is trivial. If $a' \neq a$ then we cannot compute $c(q,p)$ because $c(q)$ requires an argument of type $\Sigma x : A.Evt_A(a')$ and an element e, the second projection of p, is of type $Evt_A(a)$. As a and a' are different so are $Evt_A(a)$ and $Evt_A(a')$ and therefore $e :/ Evt_A(a')$. In the case of $a = a'$ and $e \neq e'$ we can see that $c(q)$ equals to $c(p)$ hence preserving the coherency:

$$\begin{aligned} c(q) &= c((a', e')) \\ &= c((a, e')) \\ &= (\lambda p' : (\Sigma x : A.Evt_A(a))).\pi_2(p')) = c(p) \\ &\text{with:} \\ c(q,p) &= \pi_2(p) \end{aligned}$$

□

In summary, the proposed system of coercions is coherent. However, a separate question, that we set aside for future work, is how intuitive this system is and whether a more intuitive solution can be found.

Besides, if the qualia is not of Event-type but of a type such as $\Pi h : Human.Evt_{AP}(h, Arg)$, the proposed manner of coercing to it would leave the whole expression incorrectly typed because contexts usually require one of the proper subtypes of *Event* (i.e. $Evt_A(x)$, $Evt_P(y)$, $Evt_{AP}(x, y)$), not the map from agent to event as it happens in the case we have. In other words, qualia like these require further parametrization in order to get the shape of Event-type and the paramatrization is not happening as we simply project to the object. In this case, when qualia records are intended to get parametrized by the action subject which is not the "owner" of these records, as in the case of "*John starts a book*", we would like to coerce the noun into an event-type which is already fully specified. Just the bare Σ-projection would not fulfil this need.

Dependent unit types

We now show that if we try to resolve the issue of having underparametrized qualia records via an auxiliary method, it would work as intended but once again only for the restricted case of distinct qualia types otherwise we would be reintroducing incoherence.

The auxiliary method that can be employed here is as follows: instead of having an entry $e : \Pi x : A.Evt_{AP}(e, Arg)$ in the record type, we may use a *dependent unit type* $\mathbf{1}^D_{telic\_A}(b)$ which would have an explicit parameterized coercion to the desired object.

Definition 1. *For each noun A a Dependent unit types* $\mathbf{1}^D_{telic\_A}(a)$ *and* $\mathbf{1}^D_{agentive\_A}(a)$ *are constants defined with the following definition:*

$$\mathbf{1}^D_{telic\_A},\ \mathbf{1}^D_{agentive\_A} : (A)Type$$

together with the same rules of unit types defining existence and uniqueness of their object. The same notion also can be expressed via the following inference rules:

$$\frac{\Gamma \vdash_\Delta a : A}{\Gamma \vdash_\Delta \mathbf{1}^D_{telic\_A}(a) : Type} \qquad \frac{\Gamma \vdash_\Delta \mathbf{1}^D_{telic_A}(a) : Type}{\Gamma \vdash_\Delta telic\_A_a : \mathbf{1}^D_{telic\_A}(a)}$$

$$\frac{\Gamma, z : \mathbf{1}^D_{telic\_A}(a) \vdash_\Delta C(z) : Type \quad \Gamma \vdash_\Delta c : C(telic\_A_a) \quad \Gamma \vdash_\Delta z : \mathbf{1}^D_{telic\_A}(a)}{\Gamma \vdash_\Delta \mathcal{E}_{telic\_A_a}(C, c, z) : C(z)}$$

$$\frac{\Gamma, z : \mathbf{1}^D_{telic\_A}(a) \vdash_\Delta C(z) : Type \quad \Gamma \vdash_\Delta c : C(telic\_A_a)}{\Gamma \vdash_\Delta \mathcal{E}_{telic\_book_a}(C, c, telic\_book_a) = c : C(telic\_A_a)}$$

With use of the dependent type definition we can shift the process of functional application from an object level to the coercion application level. We described earlier how a telic quale of the noun "*book*" can be typed as $\Pi h : Human.Evt_{AP}(h, b)$, where b represents an Arg field entry. There, the further application of the agent of the event is needed to form a proper meaning of the sentence and we did this through wrapper functions which compose desired objects on the computation level. Now, we can avoid using wrapper functions if we let the respective qualia record be of type $\mathbf{1}^D_{telic\_Book}(b)$ together with the family of coercions $c(b, h)$ defined in (84) below:

(84) $\mathbf{1}^D_{telic\_Book}(b) \leq_{c(b,h)} Evt_{AP}(b, h)$

An image of the coercion c from (84) can be explicitly defined to be an event $e = reading(b,h)$, where $reading(b,h) : Evt_{AP}(b,h)$ s.t. $reading(b,h,e) : Prop$ holds for the homonymic "neo-Davisonian" event predicate of shape $reading : Evt \to Prop$.

Now the record of the noun can be composed in the following way:

$$BookT = \Sigma b : Book.\Sigma e : \mathbf{1}^{D}_{telic\_Book}.\forall h : Human.reading(h,b,e) : Type$$

or

$$BookA = \Sigma b : Book.\Sigma e : \mathbf{1}^{D}_{agentive\_Book}.\forall h : Human.writing(h,b,e) : Type$$

With the corresponding coercions defined for dependent unit types as:

$$\mathbf{1}^{D}_{telic\_Book}(b) \leq_{c_T(b,h)} Evt_{AP}(b,h)$$
$$\mathbf{1}^{D}_{agentive\_Book}(b) \leq_{c_A(b,h)} Evt_{AP}(b,h)$$

The new types $BookT$ and $BookA$ (T for Telic and A for Agentive qualia) are correctly defined as we can coerce the dependent unit into the appropriate event:

$$\dfrac{\dfrac{reading(h,b) : Evt_{AP}(h,b) \to Prop \quad e : \mathbf{1}^{D}_{telic\_Book}(b) \quad \mathbf{1}^{D}_{telic\_Book}(b) \leq_{c_T(b,h)} Evt_{AP}(b,h)}{h : Human \vdash_{\Delta} reading(h,b,e) : Prop} \ldots}{\dfrac{\ldots}{\Sigma b : Book.\Sigma e : \mathbf{1}^{D}_{telic\_Book}.\forall h : Human.reading(h,b,e) : Type}}$$

Essentially the use of distinct dependent unit types for each quale of each noun allows us to regard them as corresponding events or underspecified events and finish the specifications (if they are needed) at the coercion level. This alias mechanism removes the need to define wrapper functions from the Section 5.3.2 as now we can declare projections to those units as parametrized coercions.

Unfortunately, we cannot allow ourselves to set both projections as coercions at the same time (i.e. forming the full record $BookQ$). We can coerce to projection images only in the case of all qualia entries having distinct types. If, for instance, both telic and agentive quales of the object a of a complex noun type A would be present in the record and have coercion-aliases via dependent units of the same type $Evt_A(\pi_1(a))$ (i.e. $Evt_A(Arg)$) then it would mean that the following coherency violation takes place for come context $\mathcal{C}$:

$$\mathcal{C}(a) = \mathcal{C}(c_T(\pi_1(a), \pi_{telic}(a))) = \mathcal{C}(c_Q(\pi_1(a), \pi_{agentive}(a)))$$

where π_{telic} and $\pi_{agentive}$ are corresponding Σ-projections which we set to be coercions following the definition (83). The equality can be allowed only in the case when those coercion images coincide but that is not generally the case.

Therefore, we can only choose beforehand which typing is assigned to each entry of noun like "*book*" in the phrase, either $BookT$ or $BookA$.

6 Further Research: Meta-rules

There is a possibility to parse cases of selectional coercion on a level which is outside of the MTT inference system: through *meta-rules*, as we will call them. This approach can

be linguistically sound but it may give rise to computational issues and hence make the use of proof assistants difficult or impossible.

In order to give an analysis of a phrase we need to form the MTT-judgement based on the syntactic structure of the phrase and only then we can use inference rules to check if the judgment is derivable or not. The step of translation from a syntactic structure to a semantic MTT-judgement is when we can use meta-rules i.e. where we can employ new *mechanisms of the judgement formation.* For example, adjectival modification in MTTs, as noted in the Section 3.3.2, uses Σ-types, following proposals of Mönnich 1985, Sundholm 1986 and Ranta 1995. This composition is not dictated solely by the syntactic structure of the modified noun phrase (we can see how the well-known approach of typing adjectives as $NP \to NP$ is way closer to the sole structure of a syntactic tree). Hence, maybe compositional rules from Generative Lexicon (described in the Section 4.3 of this work) can be used in the straightforward an literal way of this stage of judgement formation for example as in the following definition of the rule QP:

Qualia Projection Rule (QP): *If an interpretation $f(a)$ of the natural language phrase in question is ill-typed and a is of shape $\Sigma x : A.E$ where A is of CN universe and E is a dependent event type or is a Σ-type which can be projected to a dependent event type try interpreting the phrase as $f(p(a))$ instead, where p is the corresponding projection of type Event or its subtypes.*

In other words, we can try employing a coercion from the lexical record at the stage of judgement formation mimicking the *selective binding* rule from Generative Lexicon (Section 4.3 of this work).

The situation with underparameterized event (the ones requiring the agent to finish the event-type formation) is more complicated as the object of the sentence needs to be passed to the object of type $\Pi x : Agent.Evt_A(x)$ which lays in the qualia of the sentence subject. Rather than trying to set the appropriate meta-rule we can generalize the existent one as in the rule QP' below:

Generalized Qualia Projection Rule (QP*): *If an interpretation $f(a)$ of the natural language phrase in question is ill-typed and a is of shape $\Sigma x : A.T$ where A is of CN universe and T is either $\mathbf{1}^D_{telic\_A}(x)$ or $\mathbf{1}^D_{agent\_A}(x)$ or is a Σ-type which can be projected to either $\mathbf{1}^D_{telic\_A}(x)$ or $\mathbf{1}^D_{agent\_A}(x)$ try interpreting the phrase as $f(p(a))$ instead, where p is the corresponding projection of type $\mathbf{1}^D_{telic\_A}(x)$ or $\mathbf{1}^D_{agent\_A}(x)$*

In view of the meta-rule QP* the parameterization of underparameterized event-types would then happen automatically on the proof level via subtyping with parameterized coercion as shown in the previous section. If the qualia record does not need to be parameterized and is just of type $Event$, $Evt_A(x)$ ot $Evt_P(x)$ the meta-rule QP* still can be used with coercion parameterized only by the Arg field and not by two parameters as for underspecified events: $\mathbf{1}^D_{agent\_A} \leq_{c(x)} Evt_A(x)$.

The approach we just described is just an outline of the possible research as several questions are posed and not yet answered:

- Would employment of the rules in style of QD and QD* where different typings are checked preserve computational properties of MTTs and their implementations in proof assistants?

- Would the solution of this sort be sufficient for successful parsing of selectional coercion cases described in this work? Some of the necessary selectional coercion may be needed on the level of computation of the expression, hence not visible by the initial typing.
- As it allows more freedom would the use of such meta-rules overgenerate by yielding undesired results such as correctness of semantically absurd expression?

7 Conclusion

Let us reiterate the main questions this work addresses, posed originally in Section 1.2:

Main Questions: What are the possible ways of integrating lexical semantics from the Generative Lexicon for cases of linguistical coercion into Modern Type Theory? To what extent can selectional coercion be modelled in MTT enriched with GL-style lexical structure?

In the current work we defined an enhanced way to interpret nouns in MTTs via recording their lexical meta-properties with the nested Σ-types and dependent event types. We showed that we can model restricted cases of selectional coercion of nouns in contexts of verbal and adjective predication in two different ways: with overloading predicates to address corresponding qualia or via parametrizing secondary Σ-projections which lets us declare them as MTT-coercions. The restricted language fragment which can be analyzed is this manner should only include nouns which have qualia of strictly different types (and with disjoint sets of supertypes) in order not to introduce incoherence and further inconsistency to the calculus.

Another investigation we conducted concerned selectional coercion constructions that are ambiguous. We showed that by no means is it possible to allow different equally plausible interpretations of selectional meaning shifts (such as "to start reading a book" and "to start writing a book" induced from "to start a book") solely by means of the LF_Δ system even extended with new constructions we proposed. As discussed in Section 6, it might be worth investigating possibilities of contributing to ambiguous phrase semantics formation on a more abstract level, namely, at the stage of translation from sentence's syntactic structure to the corresponding LF_Δ judgement.

References

Asher, N. and Z. Luo (2013). "Formalization of coercions in lexical semantics". In: *Proceedings of Sinn und Bedeutung* 17, pp. 63–80.

Barwise, Jon and Robin Cooper (1981). "Generalized Quantifiers and Natural Language". In: *Linguistics and Philosophy* 4.2, pp. 159–219. (Visited on 04/19/2024).

Brasoveanu, A. (2010). *Handout 2: Two Type Logics*. URL: `https://people.ucsc.edu/~abrsvn/handout_2.pdf`.

Carnap, R. (1947). *Meaning and Necessity*. Chicago: University of Chicago Press.

Chatzikyriakidis, S. and Z. Luo (2012). "An account of natural language coordination in type theory with coercive subtyping". In: *In Proceedings of Constraint Solving and Language Processing: CSLP 2012s*, pp. 31–51.

Chatzikyriakidis, S. and Z. Luo (2013). "Adjectives in a modern type-theoretical setting with Both Strong and Weak Sums". In: *Proceedings of Formal Grammar: FG 2013, FG 2012, Morrill, G. and Nederhof, M.J. (eds). Springer, Berlin, Heidelberg*, pp. 159–174.
— (2020). *Formal Semantics in Modern Type Theories*. London: Wiley.
Chomsky, N. (1957). *Syntactic Structures*. Janua linguarum (Mouton, Paris).: Series Minor. Mouton.
Cooper, R. (2023). *From Perception to Communication: A Theory of Types for Action and Meaning*. Oxford Studies in Semantics and Pragmatics Series. Oxford University Press.
Coq (2010). *Algebraic Semantics in Language and Philosophy*. INRIA.
Curry, H. and R. Feys (1958). *Combinatory Logic, Volume 1*. Amsterdam: North Holland Publishing Company.
Davidson, D. (1967). "The logical form of action sentences". In: *Logic of Decision and Action*, pp. 216–234.
Fitting, M. (2022). "Intensional Logic". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta and Uri Nodelman. Winter 2022. Metaphysics Research Lab, Stanford University.
Frege, G. (1892). "Über Sinn und Bedutung". In: *Zeitschrift für Philosophie und philosophische Kritik* 100, pp. 25–5-.
Gallin, Daniel (1975). *Intensional and Higher-Order Modal Logic: With Applications to Montague Semantics*. New York: American Elsevier Pub. Co.
Gentzen, G. (1935). "Untersuchungen uber das logische Schließen". In: *Mathematische Zeitschrift* 39, pp. 176–210.
Howard, W.A. (1980). "The formulae-as-types notion of construction". In: *To H.B. Curry: Essays on Combinatory Logic*, pp. 469–490.
Janssen, T. M. V. and T. E. Zimmermann (2021). "Montague Semantics". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Summer 2021. Metaphysics Research Lab, Stanford University. URL: `https://plato.stanford.edu/archives/sum2021/entries/montague-semantics`.
Lambek, Joachim (1958). "The Mathematics of Sentence Structure". In: *Journal of Symbolic Logic* 65.3, pp. 154–170.
Landman, F. (n.d.). *Advanced Semantics. Chapter Five: Intensional Type Logic*. URL: `https://www.tau.ac.il/~landman/files/advanced-semantics/5%20Intensional%20Type%20Logic.pdf`.
Link, G. (1998). *Algebraic Semantics in Language and Philosophy*. CSLI Publications.
Luo, Z. (1994). *Computation and Reasoning: A Type Theory for Computer Science*. International Series of Monographs on Computer Science. Clarendon Press.
— (1997). "Coercive subtyping in type theory". In: *Computer Science Logic 1996, Lecture Notes in Computer Science 1258*.
— (1999). "Coercive subtyping". In: *Journal of Logic and Computation* 9(1), pp. 105–140.
— (2011a). "Contextual Analysis of Word Meanings in Type-Theoretical Semantics". In: *Lecture Notes in Artificial Intelligence* LNAI 6736, pp. 169–174.
— (Jan. 2011b). "Type-theoretical semantics with coercive subtyping". In: *Proceedings of SALT; Vol 20 (2010); 38-56* 20.
— (2014). "Formal Semantics in Modern Type Theories: Is It Model-Theoretic, Proof-Theoretic, or Both?" In: *Logical Aspects of Computational Linguistics 2014 (LACL 2014)*, pp. 177–188.

Luo, Z. (2019). "Donkey Anaphora: Type-Theoretic Semantics with Both Strong and Weak Sums". In: *Proceedings of the ESSLLI 2021 Workshop on Computing Semantics with Types, Frames and Related Structures, pp. 45–52.*
— (2021). "Proof Irrelevance in Type-Theoretical Semantics". In: *Logic and Algorithms in Computational Linguistics 2018 (LACompLing2018), Studies in Computational Intelligence (SCI). Springer.*
Luo, Z. and S. Soloviev (2017). "Dependent event types". In: *Logic, Language, Information, and Computation* 10388.
Luo, Z., S. Soloviev, and T. Xue (2013). "Coercive subtyping: Theory and implementation". In: *Information and Computation* 223, pp. 18–42.
Martin-Löf, P. (1984). *Intuitionistic Type Theory.* Naples: Bibliopolis.
— (1998). "An intuitionistic theory of types, Twenty-five years of constructive type theory (Venice, 1995)". In: *Oxford Logic guides* 36, pp. 127–172.
Mönnich, U. (1985). *Untersuchungen zu einer konstruktiven Semantik fur ein Fragment des Englischen.* Habilitation, University of Tübingen, Tübingen.
Montague, R. (1970). "Universal Grammar". In: *Theoria* 36, pp. 373–398.
Nunberg, Geoffrey (1979). "The Non-Uniqueness of Semantic Solutions: Polysemy". In: *Linguistics and Philosophy* 3.2, pp. 143–184.
Parsons, T. (1990). In: *Events in the Semantics of English.*
Pustejovsky, J. (1996). *The Generative Lexicon.* Cambridge: MIT Press.
— (2005). *A Survey of Dot Objects.* Waltham Mass.: Brandeis University. Ms.
— (2008). "From concepts to meaning: The role of lexical knowledge". In: *Unity and Diversity of Languages*, pp. 73–84.
Pustejovsky, J. and P. Anick (1988). "On The Semantic Interpretation Of Nominals". In: *Proceedings of COOLING-1988.*
Quine, W. V. (1956). "Universal Grammar". In: *The Journal Of Philosophy* 53, No. 5, pp. 177–187.
Ranta, A. (1995). *Type-theoretical Grammar.* Oxford: Oxford Press.
Russell, B. (1903). *The Priciples of Mathematics.* Cambridge: Cambridge University Press.
Sundholm, G. (1986). "Proof Theory and Meaning". In: *Handbook of Philosophical Logic: Volume III: Alternatives in Classical Logic.* Ed. by D. Gabbay and F. Guenthner. Dordrecht: Springer Netherlands, pp. 471–506.
— (1989). "Constructive Generalized Quantifiers". In: *Synthese* 79.1, pp. 1–12.
Wittgenstein, L. (1922). "Tractatus Logico-Philosophicus". In: *London: Routledge, 1981.*
Xue, T. (2013). *Theory and Implementation of Coercive Subtyping.* PhD thesis, Royal Holloway University of London.
Zimmermann, T. E. (1989). "Intensional Logic and Two-Sorted Type Theory". In: *The Journal of Symbolic Logic* 54.1, pp. 65–77.