Embedding Intentional Semantics into Inquisitive Semantics

Valentin D. Richard

Université de Paris

Embedding Intentional Semantics into Inquisitive Semantics

Valentin D. Richard

Master report

Université de Paris
supervised by Philippe de Groote, Sémagramme team, Loria, Inria Nancy Grand Est, France
July 7, 2021

Résumé

Plongement de la sémantique intentionnelle en sémantique inquisitrice

La sémantique inquisitrice [14] est un modèle de la sémantique de la langue qui représente uniformément les phrases interrogatives et déclaratives. Les propositions sont représentées par un ensemble d'ensembles de mondes possibles, non vide et clos par le bas, dont les éléments maximaux sont appelés alternatives. Les questions ont plusieurs alternatives, lesquelles correspondent à leurs réponses possibles. Dans ce mémoire, on examine le plongement de la sémantique intentionnelle dans la sémantique inquisitrice. On conçoit une extension conservatrice [23] qui à toute représentation sémantique lexicale associe un sens inquisiteur. On prouve que cette transformation conserve la conséquence logique (et donc l'équivalence logique) et la composition.

Abstract

Inquisitive semantics [14] is a model of natural language semantics which uniformly represents interrogative and declarative sentences. Clauses are represented by a nonempty downward-closed set of sets of possible worlds, the maximal elements of which are called alternatives. Questions have several alternatives corresponding to their possible answers. In this thesis, we investigate an embedding of (declarative) intentional semantics into inquisitive semantics. We provide a conservative extension [23] mapping every lexical meaning to an inquisitive meaning. We prove that this transformation preserves entailment (and thus logical equivalence) and composition.

Section I contains reminders about simply typed λ-calculus and its denotational semantics. Section II is dedicated to the analysis of the syntax of French questions. We model them with abstract categorial grammars and provide examples. In section III, we build an interface with Montague-style intentional semantics. Finally, we design the inquisitivation procedure in section IV and prove our main theorem.

Contents

Contents **3**

I. Preamble **5**
I.1 Introduction . . . 5
I.1.1 State of the art . . . 5
I.1.2 Goals and method . . . 5
I.1.3 Contribution and outline . . . 6
I.2 Formal tools . . . 6
I.2.1 Deduction system . . . 7
I.2.2 Relations . . . 8
I.3 Simply typed λ-calculus . . . 9
I.3.1 Simple types . . . 9
I.3.2 λ-terms . . . 10
I.3.3 Typing derivations . . . 12
I.3.4 Main properties . . . 15
I.3.5 Henkin models . . . 16

II. Syntax of French questions with ACGs **20**
II.1 Syntax of French questions . . . 20
II.1.1 Long-distance dependencies . . . 20
II.1.2 French interrogatives . . . 21
II.2 Abstract categorial grammars . . . 22
II.2.1 Definitions . . . 22
II.2.2 Order hierarchy . . . 23
II.2.3 Connection with other formalisms . . . 24
II.2.4 Additional structures on ACGs . . . 24
II.2.5 Complexity . . . 24
II.3 Syntax modelization . . . 25
II.3.1 Toy fragment of French . . . 25
II.3.2 Deep syntax . . . 26
II.3.3 Surface syntax . . . 28

III. Interface with intentional semantics **30**
III.1 Formal logic semantics . . . 30
III.1.1 Predicate semantics . . . 30
III.1.2 Montagovian semantics . . . 31
III.2 Object language . . . 33
III.2.1 Logical setting . . . 33
III.2.2 Extensional semantics . . . 35
III.3 Intentional semantics . . . 37
III.3.1 Beyond truth-conditions . . . 37
III.3.2 Logical structure on worlds . . . 38
III.3.3 Intentional lexical meanings . . . 40

IV. Extending to inquisitive semantics **42**
IV.1 Inquisitive semantics 42
IV.1.1 A representation of interrogative meanings 42
IV.1.2 First-order epistemic inquisitive logic 43
IV.1.3 Inquisitive models 45
IV.2 Conservative extensions 48
IV.2.1 De Groote's construction 48
IV.2.2 Entailment-conservative extension 50
IV.3 Inquisitivation 51
IV.3.1 Application 51
IV.3.2 Transformation of logical connectives 52
IV.3.3 Transformation of linguistic constants 54

V. Conclusion **56**

Bibliography **57**

Index **61**

I.
Preamble

I.1 Introduction

I.1.1 State of the art

Montague semantics [46] is a formalism which represents the meaning of natural language utterances with formulas based on first-order logic and λ-calculus. Montague semantics is truth-conditional: the meaning of a sentence is the set of conditions under which its formula is true. These conditions are expressed by Tarskian models.

Intentional semantics [35] improves this setting by considering a set W of possible worlds which can parameterize functions. The meaning of a sentence is the set S of possible worlds in which its formula is true. This allows attitude verbs to be modeled, especially epistemic modalities describing the knowledge of agents.

One main issue of intentional semantics is the inability to represent questions. To bridge this gap, inquisitive semantics [14] was developed recently. This semantics represents declarative and interrogative propositions on a same level.

Inquisitive meaning is based on sets of possible worlds $S \subseteq W$, called states. A formula φ is *settled* (or supported) by a state S whenever there is enough information in S to resolve φ. If φ is an interrogative meaning, settling φ amounts to be able to give an answer to φ. If φ is a declarative meaning, settling φ amounts to entail the intentional meaning of φ.

Contrary to previous attempts to model questions [34], inquisitive semantics $\mathsf{Inq_B}$ uses common set-theoretic and functional operations, e.g. intersection to denote conjunction [17]. A key feature is entailment, which, like in intentional semantic, is denoted by set inclusion.

Entailment is essential in many theories of natural language semantics [14]. It is the cornerstone of natural language inferences, the basis of discourse implicatures and is used to compute polarity items.

I.1.2 Goals and method

The goal of this thesis is to provide a transformation of intentional formulas into inquisitive formulas which preserved the original logic and composition. Such a transformation is named a conservative extension between two semantics. In our wase, we call it *inquisitivation*.

Some cases of conservative extensions have been proven to be instances of a general theorem [23]. The second goal of this thesis is to strengthen this theorem by analyzing the additional conditions sufficient to let this conservative extension preserve entailment, and see whether inquisitivation is a special case of it.

We consider formulas based on simply typed λ-calculus augmented with typed constants. Montague semantics is modeled in a higher-order signature. Abstract categorial grammars (ACGs) allow us to derive intentional and inquisitive semantics from Montague semantics as lexicons.

The aim of this dissertation is to be clear about the mathematical foundations of intentional and inquisitive semantics. To set sound basis to these formalisms, we recall the definitions and main properties of simply typed λ-calculus and its denotation (Henkin models) in detail. This is used to give complete proofs of the logical properties of inquisitivation.

I.1.3 Contribution and outline

The contribution of this thesis is threefold:

- Inquisitivation (section IV.3)
- Extension of the general theorem of conservative extensions to entailment (section IV.2.2)
- Syntactic analysis of a fragment of French questions (section II.3)

The rest of section I is a reminder of the main definitions and properties of the formal tools we use throughout this dissertation. In section I.2, we recall basic definitions about deduction systems and relations, which are useful for typing derivations and β-reduction among others. Then, in section I.3, we present simply typed λ-calculus, some properties and Henkin models. Henkin models is the formalism chosen to express denotation of λ-calculus, i.e. in order to regard formulas as set-theoretic objects. This allows us to relate to the notion of *issues*, as traditional in inquisitive logic, and to define entailment with the help of partial orders. This setting is developed in view of section IV.2, but is also used in sections III.3.2 and IV.1.3.

Section II is dedicated to the analysis of French questions. The syntactic phenomena involved with French interrogative clauses are exposed in section II.1. In section II.2, we present abstract categorial grammars (ACGs), their definitions and parsing properties. Linear ACGs are used in section II.3 to model a toy fragment of French sentences, including est-ce que interrogative clauses and declarative questions with *in situ* interrogative pronoun. We provide deep and surface syntax analyses of the example sentences.

In section III we build the higher-order signature of intentional logic. First, we recall first-order logic and Montagovian semantics in section III.1. Novice readers may begin with this gentle introduction to typed formal logic before going into the rest of this preamble. In section III.2, we define a syntax-semantics interface between the deep syntax of section II.3 and Montague semantics. In section III.3, we show how intentional semantics may be obtained from Montague semantics by a (non-linear) ACG.

Finally, section IV is the main part of this dissertation. Section IV.1 presents inquisitive semantics, the complete framework of first-order epistemic inquisitive logic and how we can simulate this logic in a higher-order signature. Conservative extensions, the theorem mentioned above and its extension to entailment are studied in section IV.2. Last but not least, section IV.3 exposes the special case of inquisitivation. We use our French fragment to exemplify our transformation. A special care is given to logical connectives, and how the inquisitivation of intentional connectives behave compared to their inquisitive correspondent.

I.2 Formal tools

The reader is assumed to be familiar with basic notions and notation of set theory.

Given a set X, we write X^* the set of words (i.e. finite lists) on X, ε the empty word and $\cdot$ word concatenation. $\mathbb{N}$ is the set of natural number.

First, we need a formal notion of trees to define simple types and derivation trees.

Definition 1 (Labeled tree)**.** *A finite rooted labeled tree $t = (\tau, L, \ell)$ is given by a set $\tau \subseteq \mathbb{N}^*$ of elements called nodes, a set L and a labeling function $\ell : \tau \to L$, verifying*

1. *If $\nu \in \tau$, then all prefixes of ν belong to τ*

2. If $\nu \cdot i \in \tau$ and $j \leq i$, then $\nu \cdot j \in \tau$

The empty word ε is called the root. A node $\nu \in \tau$ is called a leaf if $\nu \cdot 0 \notin \tau$, otherwise it is called an internal node. The daughters of a node $\nu \in \tau$ is the set of $\nu \cdot i$ which belong to τ for some $i \in \mathbb{N}$. We call arity of ν the cardinal of this set.

A subtree of (τ, L, ℓ) is a tree (τ', L, ℓ') such that there exists $\mu \in \mathbb{N}^*$ such that $\tau' = \{\nu \mid \mu \cdot \nu \in \tau\}$ and $\ell'(\nu) = \ell(\mu \cdot \nu)$. A subtree is direct if μ is of length 1.

Definition 2. *A **partial map** f from a set E to a set X is a pair (E', f') where $E' \subseteq E$ and f is a map from E' to X.*

*E' is called the **definition domain** of f and is noted* $\mathsf{dom}\, f$.

I.2.1 Deduction system

We assume we have a formal language $\mathcal{F}$ of mathematical formulas. In the following, we suppose this language is first-order logic (FOL) with some fixed predicate symbols (membership, other relations,...) and function symbols. A deduction system (or rule system) is a way of easily representing a set of dependent rules about properties on formula denotations.

Definition 3 (Deduction system)**.** *A rule instance of arity $n \in \mathbb{N}$ is an ordered pair $r = (A_1 ... A_n, A) \in \mathcal{F}^n \times \mathcal{F}$.*

*A **rule** R of arity n is a set of rule instances of arity n. A rule is a rule of some arity n.*

*A **deduction system** $\mathcal{K}$ is a finite set of rules.*

We usually represent a deduction system by the help of rule schemes. To do so, we assume we have a countably infinite set $\mathfrak{M}$ of **meta-variables**.

Remark 1. *We usually specify the rules of term deduction systems with rule schemes. We use a distinct infinite set of meta-variables. A formula A containing meta-variables intuitively denotes the set of formulas $A\sigma$ for any closing meta-variable substitution σ. We write a rule scheme $\mathfrak{R} = (A_1 ... A_n, A)$ with the hypotheses A_1,..., A_n above a bar, the conclusion A below it and the name of the rule on the right of it.*

Example 1. *Take the set $L = \{a, b\}$. We consider the set T of binary trees (i.e. trees of arity 2) labeled by L. If $t, t' \in T$ and $d \in L$, we write $d(t, t')$ the tree of root labeled by d and which direct subtrees are t and t'. The following deduction system $\mathcal{K}_{B_a}$ defines the predicate $B_a(t)$ "t has a branch of all a's".*

$$\frac{}{B_a(a)}\ \textit{Leaf} \qquad \frac{B_a(t) \qquad t' \in T}{B_a(a(t, t'))}\ \textit{Left} \qquad \frac{t \in T \qquad B_a(t')}{B_a(a(t, t'))}\ \textit{Right} \tag{I.1}$$

Here, the meta-variables are t and t'.[1] *Rule Leaf is of arity 0, so it is called an axiom.*

Definition 4 (Derivation)**.** *A **derivation** (or proof tree) of a deduction system $\mathcal{K}$ is a tree $(\pi, L \times \mathcal{K}, \ell)$ such that*

- *for every node $\nu \in \pi$, by noting $\ell(\nu) = (A, R)$ and $\ell(\nu \cdot i) = (A_i, R_i)$ for $\nu \cdot i \in \pi$, we have $(A_0 ... A_{n-1}, A) \in R$*

[1] Here, premises $t \in T$ and $t' \in T$ are taken as formulas of an unspecified FOL. However, it can be more common to see them as side conditions.

We say that a formula A is derivable if there exists a derivation which root formula is A.

Example 2. *We have that $B_a(a(a(b,a),b(a,b)))$ is derivable, and the proof of it is:*

$$\dfrac{\dfrac{b \in T \qquad \dfrac{}{B_a(a)}\,Leaf}{B_a(a(b,a))}\,Right \qquad b(a,b) \in T}{B_a(a(a(b,a),b(a,b)))}\,Left$$

However, the formulas $B_a(b(a(a,a),b))$ and $B_a(a(a,c))$ are not derivable because the root is a b and $c \notin T$ respectively.

Definition 5 (Subproof, proof-section)**.** *A (resp. direct) subproof of a proof tree D is a (resp. direct) subtree of D (see definition 1).*

A proof-section $D = (\pi', L, \ell)$ of a proof tree $D = (\pi, L, \ell)$ is a connected subset of π, i.e.

1. *$\pi' \subseteq \pi$*
2. *If $\mu \in \pi'$ and $\mu \cdot \nu \in \pi'$, then for all prefixes ν' of ν, $\mu \cdot \nu' \in \pi'$*
3. *ℓ' is ℓ restricted to π'*

Remark 2. *It is common to use vertical dots to declare some proof-section of a proof π. For example, the derivation of example 2 is of the form*

$$\dfrac{\dfrac{}{B_a(a)}\,Leaf \qquad b(a,b) \in T}{B_a(a(a(b,a),b(a,b)))}\,Left$$

where π' is a proof-section.

I.2.2 Relations

Definition 6. *A relation R on a set E is a subset of the cartesian product $E \times E$. The notation $a\ R\ b$ stands for $(a,b) \in R$.*

We define common properties a relation can have.

Definition 7. *A relation R on E is*

reflexive *if for all $a \in E$, $a\ R\ a$*

symmetric *if for all $a, b \in E$, if $a\ R\ b$ then $b\ R\ a$*

antisymmetric *if for all $a, b \in E$, if $a\ R\ b$ and $b\ R\ a$, then $a = b$*

transitive *if for all $a, b, c \in E$, if $a\ R\ b$ and $b\ R\ c$, then $a\ R\ c$*

Definition 8. *Set R, R' two relations on E. The composition of R and R' defined by $a\ RR'\ b$ iff there exists $c \in E$ such that $a\ R\ c$ and $c\ R'\ b$.*

The composition of relations is associative, and we write R^n for $\underbrace{R...R}_{n}$.

Definition 9. *Set R a relation on E.*

The opposite relation of R is $R^{-1} \hat{=} \{(b,a) \mid (a,b) \in R\}$.

The reflexive transitive closure of R is the relation R^* defined by the following deduction system:

$$\frac{}{a\ R^*\ a}\ \textit{Refl.} \quad \frac{a\ R\ b}{a\ R^*\ b}\ \textit{Incl.} \quad \frac{a\ R^*\ b \qquad b\ R^*\ c}{a\ R^*\ c}\ \textit{Trans.} \tag{I.2}$$

The equivalence relation $=_R$ generated by R is $=_R \hat{=}\ (R \cup R^{-1})^*$.

I.3 Simply typed λ-calculus

λ-calculus is a useful compact notation to write functions. It was invented in the 1930s by Alonzo Church. Simply typed λ-calculus is a well-behaved fragment of λ which is sufficient to express simple functions with no recursion. We refer to [4] for more details on λ-calculus.

We begin by defining simple types in section I.3.1. Then we define λ-calculus and linear λ-calculus in section I.3 and typing derivations in section I.3.3. After some recap of main properties of β- and η-reduction in section I.3.4, we finally give a formal setting of Henkin models in section I.3.5.

I.3.1 Simple types

Definition 10 (Ranked alphabet)**.** *A ranked alphabet is a finite set Σ of elements called symbols together with an arity function $\mathsf{ar} : \Sigma \to \mathbb{N}$.*

For $n \in \mathbb{N}$, we write Σ_n the subset of Σ containing the symbols of arity n.

To talk about uniform substitution, we require a (countably) infinite set $\mathcal{V}$ of elements called type **type variables**.

Definition 11 (Term)**.** *A term t on a set $\mathcal{V}$ and a ranked alphabet Σ is a nonempty finite rooted labeled tree (τ, Σ, ℓ) satisfying, for all node $\nu \in \tau$,*

- *ν has n daughters iff $\mathsf{ar}(\ell(\nu)) = n$*
- *if $\ell(\nu) \in \mathcal{V}$, then ν is a leaf*

*When a term contains no type variable, we call it a **ground term**. We write $\mathcal{T}(\Sigma, \mathcal{V})$ the set of terms on Σ and $\mathcal{T}(\Sigma)$ the set of ground terms.*

Remark 3. *We usually define terms by the following induction scheme:*

$$A ::= \delta \in \mathcal{V} \mid f(A_1, ..., A_n), f \in \Sigma_n \tag{I.3}$$

Definition 12 (Uniform substitution)**.** *Set Σ a ranked alphabet. A uniform substitution is a partial map σ from type variables $\mathcal{V}$ to terms $\mathcal{T}(\Sigma, \mathcal{V})$.*

Applying a uniform substitution σ to a term A gives $A\sigma \in \mathcal{T}(\Sigma)$ constructed by replacing simultaneously every $\delta \in \mathcal{V}$ occurring in A by $\sigma(\delta)$ if it is defined. Equivalently, we have by induction

$$\begin{array}{rcll} \delta\sigma & = & \sigma(b) & \text{if } \delta \in \mathsf{dom}\,\sigma \\ \delta\sigma & = & b & \text{if } \delta \notin \mathsf{dom}\,\sigma \\ f(A_1, ..., A_n)\sigma & = & f(A_1\sigma, ..., A_n\sigma) & \end{array} \tag{I.4}$$

We say that σ is grounding if for any $\delta \in \mathsf{dom}\,\sigma$, $\sigma(\delta) \in \mathcal{T}(\Sigma)$.

For example, set $\mathcal{B} = \{b, f\}$ with $\mathsf{ar}(a) = \mathsf{ar}(b) = 0$ and $\mathsf{ar}(f) = 2$. Then $\sigma = [\delta \mapsto f(\gamma, b)]$ is a substitution, and $f(\gamma, f(f(b, \delta), \delta))\sigma = f(\gamma, f(f(b, f(a, \gamma)), f(a, \gamma)))$.

Definition 13 (Simple types)**.** *Let $\mathcal{B}$ be a finite set, the elements of which are called atomic types. The set $\mathcal{T}(\mathcal{B}, \mathcal{V})$ of **simple types** is the set of terms build on the ranked alphabet $\{\to\} \cup \mathcal{B}$ and type variables $\mathcal{V}$, with $\mathsf{ar}(\to) = 2$ and $\mathsf{ar}(b) = 0$ if $b \in \mathcal{B}$. In other words:*

$$\mathcal{T}(\mathcal{B}) \ni A, B ::= \delta \in \mathcal{V} \mid b \in \mathcal{B} \mid A \to B \tag{I.5}$$

We commonly put the arrow between its two arguments (infix notation).

We write $\mathcal{T}(\mathcal{B})$ the set of ground types, i.e. simple types with no type variable.

By abuse of language, we often write "type" instead of "ground type" in the rest of this dissertation. Non ground types are only used in sections I.3.3 and I.3.5.

Instead of writing all parentheses, we use the conventional notation $A \to B \to C$ to mean $A \to (B \to C)$.

Remark 4. *As types may be very long, to gain space we may omit the arrow on types. For example, $A(AB)B$ stands for $A \to ((A \to B) \to B)$*

I.3.2 λ-terms

Definition 14 (Higher-order signature)**.** *A **higher-order signature (HOS)** is a triplet $\Sigma = (\mathcal{B}, C, \mathtt{t})$ where*

- *$\mathcal{B}$ is a finite set of atomic types*
- *$\mathcal{C}$ is a finite set of constants*
- *$\mathtt{t} : C \to \mathcal{T}(\mathcal{B})$ assigns a ground type to every constant*

Remark 5. *Here, the word "constant" does not refer to first-order symbols of arity 0, but to any symbol of any type.*

Let $\mathcal{X}$ be a countably infinite set. We call the elements of $\mathcal{X}$ **λ-variables** $x, y, z, \ldots$. We fix this set for the rest of the paper and we assume that it is disjoint with every set of constants considered here.

A **λ-term** can be constructed with constants, variables and two rules: 1. application $f\,a$ applies a function f to its argument a; 2. λ-abstraction $\lambda a.\, f(a)$ creates a function $a \mapsto f(a)$ out of an expression $f(a)$.

Definition 15 (λ-term)**.** *The set of λ-terms $\Lambda(\Sigma)$ on a HOS $\Sigma = (\mathcal{B}, \mathcal{C}, \mathtt{t})$ is defined inductively by the following deduction system:*

$$\frac{c \in C}{c \in \Lambda(\Sigma)}\ \textit{Const.} \qquad \frac{x \in \mathcal{X}}{x \in \Lambda(\Sigma)}\ \textit{Var.}$$
$$\frac{M \in \Lambda(\Sigma) \qquad x \in \mathcal{X}}{\lambda x.\, M \in \Lambda(\Sigma)}\ \lambda\textit{-abs.} \qquad \frac{M \in \Lambda(\Sigma) \qquad N \in \Lambda(\Sigma)}{M\,N \in \Lambda(\Sigma)}\ \textit{App.} \tag{I.6}$$

Remark 6. *We give priority to application over λ-abstraction, e.g. $\lambda x.\, y\, x$ means $\lambda x.\, (y\, x)$. Similarly to types, we conventionally write $M\, N_1\, N_2$ for $(M\, N_1)\, N_2$. The intuition behind that is that M acts like a function of two argument N_1 and N_2.*

It is also common to group λ-abstracted variables like this $\lambda x, y.\, M$ instead of $\lambda x.\, \lambda y.\, M$.

For example, $M = \lambda x.\, x\,(c\,y)$ and $N = \lambda x.\, \lambda x.\, x$ are λ-terms. Like functions, we intuitively want to be able to substitute a λ-abstracted variable by another one if we also change the occurrences of this variable in the rest of the λ-term. For instance, M would be equivalent to $\lambda z.\, z\,(c\,y)$. However, replacing variables naively could fail to implement what we actually mean: N should not be equivalent to $\lambda z.\, \lambda x.\, z$. A solution to this issue was given by Church. Here, we give the formulation of [33].

Definition 16 (Free and bound variables)**.** *The set of* ***free variables*** $\mathsf{FV}(M)$ *and* ***bound variables*** $\mathsf{BV}(M)$ *of a λ-term M is defined inductively on M by:*

$$\begin{array}{llll} \mathsf{FV}(c) \mathrel{\hat{=}} \emptyset & \mathsf{FV}(x) \mathrel{\hat{=}} \{x\} & \mathsf{FV}(\lambda x.\, M) \mathrel{\hat{=}} \mathsf{FV}(M) \setminus \{x\} & \mathsf{FV}(M\,N) \mathrel{\hat{=}} \mathsf{FV}(M) \cup \mathsf{FV}(N) \\ \mathsf{BV}(c) \mathrel{\hat{=}} \emptyset & \mathsf{BV}(x) \mathrel{\hat{=}} \emptyset & \mathsf{BV}(\lambda x.\, M) \mathrel{\hat{=}} \mathsf{BV}(M) \cup \{x\} & \mathsf{BV}(M\,N) \mathrel{\hat{=}} \mathsf{BV}(M) \cup \mathsf{BV}(N) \end{array} \tag{I.7}$$

M is ***closed*** *when $\mathsf{FV}(M) = \emptyset$.*

Definition 17 (Substitution)**.** *A variable x is substitutable by N in M if $x \notin \mathsf{BV}(M)$ and $\mathsf{FV}(N) \cap \mathsf{BV}(M) = \emptyset$. In this case, we define the* ***substitution*** *$M[x := N]$ by induction on M:*

$$\begin{array}{rcll} x[x := N] & \mathrel{\hat{=}} & N & \\ y[x := N] & \mathrel{\hat{=}} & y & \text{if } x \neq y \\ c[x := N] & \mathrel{\hat{=}} & c & \\ (M_1\,M_2)[x := N] & \mathrel{\hat{=}} & (M_1[x := N])\,(M_2[x := N]) & \\ (\lambda y.\, M)[x := N] & \mathrel{\hat{=}} & \lambda y.\, (M[x := N]) & \end{array} \tag{I.8}$$

For example, $(\lambda x.\, c\,(y\,y))[y := \lambda z.\, c] = \lambda x.\, c\,((\lambda z.\, c)\,(\lambda z.\, c))$.

Definition 18 (Contextualization)**.** *Given a relation R on $\Lambda(\Sigma)$, the contextualization R' of R is defined by the following deduction system:*

$$\frac{M_1\ R\ M_2}{M_1\ R'\ M_2} \qquad \frac{M_1\ R'\ M_2}{M_1\,N\ R'\ M_2\,N} \qquad \frac{N_1\ R'\ N_2}{M\,N_1\ R'\ M\,N_2} \qquad \frac{M_1\ R'\ M_2}{\lambda x.\, M_1\ R'\ \lambda x.\, M_2} \tag{I.9}$$

Definition 19 (α-equivalence)**.** *α-****renaming*** *is the contextualized relation obtained from*

$$\lambda x.\, M \quad \alpha \quad \lambda y.\, (M[x := y]) \tag{I.10}$$

for every y such that x is subtitutable by y in M and y is not free in M (i.e. $y \notin \mathsf{FV}(M)$).

*α-****equivalence*** *is the equivalence relation $=_\alpha$ generated by α-renaming.*

For example, $\lambda x.\, (\lambda y.\, x\,y)\,c =_\alpha \lambda z.\, (\lambda x.\, z\,x)\,c$.

Remark 7. *In the following, we always assume that for every M, $\mathsf{FV}(M) \cap \mathsf{BV}(M) = \emptyset$, or that we automatically α-rename M to obtain these property in order to substitute correctly. By* ***fresh variable****, we mean a variable x which does not occur freely in the considered λ-terms or typing judgments.*

Linear λ-terms

We write $\mathcal{T}^\circ(\mathcal{B})$ the set of simple types with the different arrow $\multimap$ and call them **linear types**.

Definition 20. *The set of **linear λ-terms** $\Lambda^\circ(\Sigma)$ on a linear HOS Σ (i.e. $\mathtt{t} : C \to \mathcal{T}^\circ(\mathcal{B})$) is defined as the subset of $\Lambda(\Sigma)$ satisfying the additional constraints:*

- *for every constant $c \in C$, $c \in \Lambda^\circ(\Sigma)$*
- *for every variable $x \in \mathcal{X}$, $x \in \Lambda^\circ(\Sigma)$*
- *for every $M \in \Lambda^\circ(\Sigma)$ and variable $x \in \mathcal{X}$ such that x occurs freely exactly once in M, $\lambda^\circ x.\, M \in \Lambda^\circ(\Sigma)$*
- *for every $M, N \in \Lambda^\circ(\Sigma)$ such that $\mathsf{FV}(M) \cap \mathsf{FV}(N) = \emptyset$, $M\, N \in \Lambda^\circ(\Sigma)$*

The notation λ° is just there to distinguish linear λ-terms from regular λ-terms.

For example, $\lambda^\circ x, y.\, c\,(c\,(x\,y))$ is linear, but $x\,(\lambda c.\, x)$ is not linear.

Lemma 1. *If $M, N \in \Lambda^\circ(\Sigma)$ and $(\mathsf{FV}(M) \cup \mathsf{BV}(M)) \cap \mathsf{FV}(N) = \emptyset$, then $M[x := N] \in \Lambda^\circ(\Sigma)$.*

Linear λ-terms are closed under α-equivalence, i.e. if M is linear and $N =_\alpha M$, then N is linear.

I.3.3 Typing derivations

We can associate one or several simple types to some λ-terms [32]. The proof that a λ-term has a certain type is formalized as a typing derivation. In this subsection, we fix a HOS Σ.

Definition 21 (Typing judgment)**.** *A (resp. linear) typing declaration $x : A$ is formally a couple (x, A) where $x \in \mathcal{X}$ and $A \in \mathcal{T}(\mathcal{B})$ (resp. $A \in \mathcal{T}^\circ(\mathcal{B})$).*

A (resp. linear) typing environment is a list Γ of (resp. linear) typing declarations. The empty typing environment is represented by an blank space.

A (resp. linear) typing judgment is a triplet (Γ, M, A) written $\Gamma \vdash M : A$, where Γ is a (resp. linear) typing context, $M \in \Lambda(\Sigma)$ (resp. $M \in \Lambda^\circ(\Sigma)$) and $A \in \mathcal{T}(\mathcal{B})$ (resp. $A \in \mathcal{T}^\circ(\mathcal{B})$).

We define $\mathsf{DV}(\Gamma)$ as the set of variables involved in the typing declarations of Γ.

Note that here, the turnstile $\vdash$ is a mere notation.

Definition 22 (Typing derivation)**.** *A **typing derivation** (or proof) is a derivation of the following deduction system:*

$$\frac{\mathtt{t}(c) = A}{\Gamma \vdash c : A}\ \textit{Const.} \qquad \frac{x : A \in \Gamma}{\Gamma \vdash x : A}\ \textit{Var.}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.\, M : A \to B}\ \lambda\textit{-abs.} \qquad \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\, N : B}\ \textit{App.} \tag{I.11}$$

$$\frac{\Gamma, x : C, y : B, \Delta \vdash M : A}{\Gamma, y : B, x : C, \Delta \vdash M : A}\ \textit{Exch.}$$

Where a comma denotes concatenation of typing environments.

Definition 23 (Linear typing derivation)**.** *A linear typing derivation is a derivation of the following deduction systems on linear typing judgments:*

$$\frac{\mathtt{t}(c) = A}{\vdash c : A}\ \textit{Const.} \qquad \frac{}{x : A \vdash x : A}\ \textit{Var.}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda^{\circ} x.\, M : A \multimap B}\ \lambda\textit{-abs.} \qquad \frac{\Gamma \vdash M : A \multimap B \qquad \Delta \vdash N : A}{\Gamma, \Delta \vdash M\, N : B}\ \textit{App.} \tag{I.12}$$

$$\frac{\Gamma, x : C, y : B, \Delta \vdash M : A}{\Gamma, y : B, x : C, \Delta \vdash M : A}\ \textit{Exch.}$$

Where a comma denotes concatenation of typing environments. Note that with the conditions on linear typing judgments, in the App. rule we must have $\mathsf{DV}(\Gamma) \cap \mathsf{DV}(\Delta) = \emptyset$*.*

Remark 8. *In the system we present here, declaring a variable multiple times (even with different types) is forbidden so that rule Var. stays consistent. Such a situation can be avoided by applying remark 7 on every sub-λ-term of the λ-term to type.*

Example 3. *With atomic types* $\mathcal{B} = \{a, b\}$ *and the following constants*

$$\begin{array}{llll} c_1 : & b \to b \to b & c_2 : & ((a \to b) \to b) \to b \\ c_3 : & a \multimap a \multimap b & c_4 : & (a \multimap b) \multimap b \end{array} \qquad c_5 : \quad a \tag{I.13}$$

we can derive the regular λ-term $M = \lambda x.\, c_2\,(\lambda y.\, c_1\,(z\,y)\,(y\,x))$ *and the closed linear λ-term* $c_4\,(\lambda^{\circ} x.\, c_3\,x\,c_5)$*. We set* $\Gamma = z : (ab)b, x : a, y : ab$*. We omit exchange rules and axioms, as usual.*

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash c_1 : bbb \qquad \dfrac{\Gamma \vdash z : (ab)b \qquad \Gamma \vdash y : ab}{\Gamma \vdash z\,y : b}\ \textit{App.}}{\Gamma \vdash c_1\,(z\,y) : bb}\ \textit{App.} \qquad \dfrac{\Gamma \vdash y : ab \qquad \Gamma \vdash x : a}{\Gamma \vdash y\,x : b}\ \textit{App.}}{\Gamma \vdash c_1\,(z\,y)\,(y\,x) : b}\ \textit{App.}}{z : (ab)b, x : a \vdash \lambda y.\, c_1\,(z\,y)\,(y\,x) : (ab)b}\ \lambda\textit{-abs.}$$
$$\dfrac{z : (ab)b, x : a \vdash c_2 : ((ab)b)b \qquad z : (ab)b, x : a \vdash \lambda y.\, c_1\,(z\,y)\,(y\,x) : (ab)b}{z : (ab)b, x : a \vdash c_2\,(\lambda y.\, c_1\,(z\,y)\,(y\,x)) : b}\ \textit{App.}$$
$$\dfrac{z : (ab)b, x : a \vdash c_2\,(\lambda y.\, c_1\,(z\,y)\,(y\,x)) : b}{z : (ab)b \vdash M : ab}\ \lambda\textit{-abs.}$$

$$\dfrac{\dfrac{\dfrac{\vdash c_3 : aab \qquad x : a \vdash x : a}{x : a \vdash c_3\,x : ab}\ \textit{App.} \qquad \vdash c_5 : a}{x : a \vdash c_3\,x\,c_5 : b}\ \textit{App.}}{\vdash \lambda^{\circ} x.\, c_3\,x\,c_5 : ab}\ \lambda\textit{-abs.}$$
$$\dfrac{\vdash c_4 : (ab)b \qquad \vdash \lambda^{\circ} x.\, c_3\,x\,c_5 : ab}{\vdash c_4\,(\lambda^{\circ} x.\, c_3\,x\,c_5) : b}\ \textit{App.}$$

In the following, we write $\Gamma \vdash_{\Sigma} M : A$ to say that $\Gamma \vdash M : A$ is **derivable**, or linearly derivable if Σ is linear (i.e. has a linear typing derivation). If $\Gamma \vdash_{\Sigma} M : A$, we say that A is a type of M (in the typing environment Γ).

We say that M is (resp. linearly) typable is there exists a (resp. linear) type A and a (resp. linear) typing environment Γ such that $\Gamma \vdash_{\Sigma} M : A$.

Proposition 1. *M is closed and $\Gamma \vdash_{\Sigma} M : A$ iff $\vdash_{\Sigma} M : A$.*

Proof. Suppose M is closed and π is a derivation of $\Gamma \vdash M : A$. We can α-rename M to get $\mathsf{DV}(\Gamma) \cap \mathsf{BV}(M) = \emptyset$. Thus, no axiom of π involves a typing declaration of Γ, because $\mathsf{FV}(M) = \emptyset$. Therefore we could globally remove the typing declarations of Γ in all typing judgment of π to get a proof of $\vdash M : A$.

The converse implication is straightforward. □

Definition 24. *We write $\Lambda(\Sigma)^A_\Gamma$ the subset of $\Lambda(\Sigma)$ of λ-terms of ground type A in the typing context Γ.*

Proposition 2. *If M is linear and derivable iff M is linearly derivable.*

Proof. Suppose M is linear and has a derivation π of root typing judgment $\Gamma \vdash M : A$. Let us build a linear derivation π' of M by induction on π.

Case Var. We have $M = x$. As $x : A \in \Gamma$, $x : a \vdash x : A$ is derivable.

Case Const. Taking $\pi' = \pi$ works.

Case λ-abs. By induction hypothesis, the direct subproof π_0 of root typing judgment $\Gamma, x : A \vdash M' : B$ admits a linear proof π'_0 because M' is linear. So λ-abs.(π'_0) is a linear proof of M

Case App. We have $M = M'\,N'$. Take Γ' (resp. Δ') the sublist of Γ such that $\mathsf{DV}(\Gamma') = \mathsf{FV}(M')$ (resp. $\mathsf{DV}(\Delta') = \mathsf{FV}(N')$). By linearity, Γ' and Δ' do not share any typing declaration. By using the induction hypothesis on $\Gamma' \vdash M' : C \multimap A$ (resp. $\Delta' \vdash N' : C$) (M' and N' are also linear), we get a linear derivation π_1 (reps. π_2). Then, $\pi' = \text{App.}(\pi_1, \pi_2)$ is a linear derivation of M.

Case Exch. By straightforwardly using the induction hypothesis on the direct subproof of π.

The reverse implication follows from the definition of linear derivations. □

Remark 9. *When declaring closed λ-terms, we use Church notation and specify one type of λ-abstracted variables, e.g. $\lambda f^{aab}, x^a.\, f\,x\,x$ means $\vdash_\Sigma \lambda f, x.\, f\,x\,x : (a \to a \to b) \to a \to b$.*

Principal derivation

Here we use non ground types to define principal derivations. This notion will be useful to prove properties about denotation of typing derivations in the following subsection.

The following results are extracted from [37] and [49].

Definition 25 (Typing derivation scheme)**.** *A typing derivation scheme is a typing derivation on a signature Σ but where in every judgment $x_1 : A_1, ..., x_n : A_n \vdash M : A$, we can have $A_1, ..., A_n, A \in \mathcal{T}(\mathcal{B}, \mathcal{V})$ instead of being ground types. The rules are the same.*

We write $\Gamma \Vdash M : A$ to indicate that the typing judgment may contain type variables, and $\Gamma \Vdash_\Sigma M : A$ if it is derivable.

Definition 26 (Principal type)**.** *A principal type of M is a type $A \in \mathcal{T}(\mathcal{B}, \mathcal{V})$ such that*

- *there exists Γ such that $\Gamma \Vdash_\Sigma M : A$*
- *if $\Gamma' \Vdash_\Sigma M : B$, there exists a uniform substitution σ such that $B = A\sigma$ and $\Gamma' = \Gamma\Sigma$*

Definition 27 (Global substitution). *If π is a typing derivation scheme and σ a uniform substitution of types, then applying σ to π is $\pi\sigma$, defined as the derivation where every typing context $x_1 : A_1, ..., x_n : A_n \Vdash M : A$ is replaced by $x_1 : A_1\sigma, ..., x_n : A_n\sigma \Vdash M : A\sigma$.*

$\pi\sigma$ is still a typing derivation scheme, and is a typing derivation if σ is grounding.

Definition 28 (Principal derivation). *A principal derivation of M is a typing derivation scheme π of root $\Gamma \Vdash M : A$ such that every other typing derivation scheme of M is obtained by globally applying a uniform substitution σ and adding no or some exchange rules.*

Proposition 3. *If M is typable, then M admits a principal derivation and a principal type (the type of M in its principal derivation).*

I.3.4 Main properties

λ-terms can sometimes be simplified and have a normal form. This is formalized by β-reduction.

Definition 29 (β-reduction). *β-**reduction** $\to_\beta$ is the contextualized relation obtained from*

$$(\lambda x.\, M)\, N \quad \beta \quad M[x := N] \tag{I.14}$$

We write $\to_\beta^*$ its reflexive transitive closure, and $=_\beta$ its equivalence relation.

Proposition 4 (Subject reduction). *If $M \to_\beta N$ and $\Gamma \vdash_\Sigma M : A$, then $\Gamma \vdash_\Sigma N : A$.*

A well-known result about (untyped) λ-calculus is the following.

Theorem 1 (Confluence). *If $M \to_\beta^* M_1$ and $M \to_\beta^* M_2$, then there exists N such that $M_1 \to_\beta^* N$ and $M_2 \to_\beta^* N$.*

Definition 30 (Strong normalization). *A λ-term M is β-normal if for every N, it is false that $M \to_\beta N$.*

M is strongly normalizing if there exists no infinite sequence $(M_n)_{n\in\mathbb{N}}$ such that $M_0 = M$ and for all $n \in \mathbb{N}$, $M_n \to_\beta M_{n+1}$.

The following theorem was proved by Tait in 1967.

Theorem 2 (Strong normalization). *If M is typable, then M is strongly normalizing.*

η-reduction is another common relation on λ-terms which characterizes extentionallity of denotations.

Definition 31 (η-reduction). *η-reduction $\to_\eta$ is the contextualized relation obtained from*

$$\lambda x.\, M\, x \quad \eta \quad M \qquad \text{if } x \notin \mathsf{FV}(M) \tag{I.15}$$

*We write $=_\eta$ its equivalence relation and $=_{\beta\eta} \,\hat{=}\, =_\beta \cup =_\eta$. η-**expansion** is the relation $\to_\eta^{-1}$*

Straightforwardly, η-reduction is strongly normalizing on every λ-term. Subject reduction also holds for η-reduction.

Note that these properties naturally extend to non ground typed λ-terms.

I.3.5 Henkin models

Henkin models [36] are a class of sound and complete models for λ-calculus. Here we give an adapted version of [19].

Definition 32 (Domain family)**.** *Set $\Sigma = (\mathcal{B}, C, \mathtt{t})$ a higher-order signature. An extensional domain family on Σ is a family of sets $(D_A)_{A\in\mathcal{T}(\mathcal{B})}$ such that for every types A, B, $D_{A\to B}$ is a subset of the set $D_B^{D_A}$ all the functions from D_A to D_B.*

Definition 33 (Assignment)**.** *Given a domain family $(D_A)_{A\in\mathcal{T}(\mathcal{B})}$ an **assignment** g is a partial function mapping some typing declaration $x : A$ to an element of D_A such that if $x : A, x : B \in \mathsf{dom}\, g$, then $A = B$.*

We write $g[x : A \mapsto a]$ the assignment g but with adding (or changing) $g(x : A)$ to equal a, viz.

$$g[x : A \mapsto a](y : B) = \begin{cases} a & \text{if } y : B \;=\; x : A \\ g(y : B) & \text{if } y : B \in \mathsf{dom}\, g \text{ and } y : B \;\neq\; x : A \end{cases} \tag{I.16}$$

Definition 34 (Henkin model)**.** *An extensional applicative structure $\mathcal{M}$ of a higher-order signature Σ is an extensional domain family $(D_A)_{A\in\mathcal{T}(\mathcal{B})}$ equipped with*

1. *a denotation $c^{\mathcal{M}} \in D_{\mathtt{t}(c)}$ of every constant $c \in \mathcal{C}$*
2. *a denotation function $[\![\cdot]\!]^{\mathcal{M},g}$ mapping a derivation π of root typing judgment $x_1 : A_1, ..., x_n : A_n \vdash M : A$ such that $\{x_1 : A_1, ..., x_n : A_n\} \subseteq \mathsf{dom}\, g$ to an element of D_A.*

*A **extensional Henkin model** is an extensional applicative structure $\mathcal{M}$ such that*

1. $[\![\Gamma \vdash x : A]\!]^{\mathcal{M},g} = g(x : A)$
2. $[\![\Gamma \vdash c : A]\!]^{\mathcal{M},g} = c^{\mathcal{M}}$
3. $[\![\Gamma \vdash M\, N : B]\!]^{\mathcal{M},g} = [\![\Gamma \vdash M : A \to B]\!]^{\mathcal{M},g}([\![\Gamma \vdash N : A]\!]^{\mathcal{M},g})$ *if*

$$\pi = \dfrac{\begin{matrix}\vdots\,\pi_1 & \vdots\,\pi_2\end{matrix}}{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\scriptstyle App.}$$
$$\Gamma \vdash M\, N : B$$

4. $[\![\Gamma \vdash \lambda x.\, M : A \to B]\!]^{\mathcal{M},g}(a) = [\![\Gamma, x : A \vdash M : B]\!]^{\mathcal{M},g[x:A\mapsto a]}$ *for every $a \in D_A$*
5. $[\![\Gamma, x : C, y : B, \Delta \vdash M : A]\!]^{\mathcal{M},g} = [\![\Gamma, y : B, x : C, \Delta \vdash M : A]\!]^{\mathcal{M},g}$

Key properties

The following propositions establish the soundness of β-reduction and η-reduction. Proofs can be found in [2].

Proposition 5. *If $M \to_\beta^* N$ and $\Gamma \vdash_\Sigma M : A$, then $[\![\Gamma \vdash M : A]\!]^{\mathcal{M},g} = [\![\Gamma \vdash N : A]\!]^{\mathcal{M},g}$ for every $\mathcal{M}, g$.*

Proposition 6. *If $M =_{\beta\eta} N$, $\Gamma \vdash_\Sigma N : A$ and $\Gamma \vdash_\Sigma M : A$, then $[\![\Gamma \vdash M : A]\!]^{\mathcal{M},g} = [\![\Gamma \vdash N : A]\!]^{\mathcal{M},g}$ for every $\mathcal{M}, g$.*

In the definition of Henkin models, by abuse of notation we only put the root typing judgment inside the $[\![\cdot]\!]^{\mathcal{M},g}$ brackets instead of π, by convenience. This notation turns out to be relevant because the denotation of a derivation is entirely determined by its root. We prove it in proposition 7.

Definition 35. *We define the denotation of a typing derivation scheme π in a global substitution σ as $[\![\pi]\!]^{\mathcal{M},g}_{\sigma} = [\![\pi\sigma]\!]^{\mathcal{M},g}$.*

If π is a typing derivation, this definition coincides with the previous one because for every global substitution σ, $\pi\sigma = \pi$.

Lemma 2. *If σ is a uniform substitution and π, π' tying derivation schemes such that $\pi' = \pi\sigma$, then for any global substitution μ and $\mathcal{M}, g$, $[\![\pi']\!]^{\mathcal{M},g}_{\mu} = [\![\pi]\!]^{\mathcal{M},g}_{\sigma\mu}$, where $(\sigma\mu)(\delta) = \sigma(\delta)\mu$ iff $\delta \in \mathsf{dom}\,\sigma$.*

Proof. We have $[\![\pi']\!]^{\mathcal{M},g}_{\mu} = [\![(\pi\sigma)\mu]\!]^{\mathcal{M},g}$. Moreover, for any $A \in \mathcal{T}(\mathcal{B}, \mathcal{V})$, it is straightforward to prove by a quick induction that $(A\sigma)\mu = A(\sigma\mu)$. So $(\pi\sigma)\mu = \pi(\sigma\mu)$ and so $[\![\pi']\!]^{\mathcal{M},g}_{\mu} = [\![\pi]\!]^{\mathcal{M},g}_{\sigma\mu}$ □

Lemma 3. *Let π_1 and π_2 be typing derivation schemes of root $\Gamma_1 \Vdash M_1 : A_1$ and $\Gamma_2 \Vdash M_2 : A_2$ respectively.*

For any common global substitution μ such that $(\Gamma_1 \Vdash M_1 : A_1)\mu = (\Gamma \Vdash M : A)\mu$, we have $[\![\pi_1]\!]^{\mathcal{M},g}_{\mu} = [\![\pi_2]\!]^{\mathcal{M},g}_{\mu}$

Proof. Using the notation of the lemma, note $\Gamma \Vdash M : A = (\Gamma_1 \Vdash M_1 : A_1)\mu$. By proposition 3, $\Gamma \Vdash M : A$ admits a principal derivation π, such that $\pi_1 = \pi\sigma_1$ and $\pi_2 = \pi\sigma_2$ up to exchange rules. We proceed by induction on the typing derivation scheme π.

Exchange rules do not change the denotation.

By lemma 2 we have $[\![\pi_1]\!]^{\mathcal{M},g}_{\mu} = [\![\pi]\!]^{\mathcal{M},g}_{\sigma\mu}$ and $[\![\pi_2]\!]^{\mathcal{M},g}_{\mu} = [\![\pi]\!]^{\mathcal{M},g}_{\sigma\mu}$.

What remains is to prove that

$$[\![\pi]\!]^{\mathcal{M},g}_{\sigma} = [\![\pi]\!]^{\mathcal{M},g}_{\sigma'} \qquad (*)$$

for any global substitutions σ, σ' agreeing on the root, and $\mathcal{M}, g$.

First, we take M' the β-normal term such that $M \to^*_\beta M'$ by theorem 2. By proposition 5, we have $[\![\Gamma \Vdash M : A]\!]^{\mathcal{M},g}_{\sigma} = [\![\Gamma \Vdash M' : A]\!]^{\mathcal{M},g}_{\sigma}$ and $[\![\Gamma \Vdash M : A]\!]^{\mathcal{M},g}_{\sigma'} = [\![\Gamma \Vdash M' : A]\!]^{\mathcal{M},g}_{\sigma'}$.

Case Var. If $\pi = \Gamma \vdash x : A$, then $\pi\sigma = \pi\sigma'$ and so $(*)$ holds.

Case Const. If $\pi = \Gamma \vdash c : A$, then $\pi\sigma = \pi\sigma'$ and so $(*)$ holds.

Case λ-abs. If $M' = \lambda x.\, M''$ and $A = B \to C$, then by applying the induction hypothesis on the direct subproof π' of π allows us to deduce, for any $b \in D_B$,

$$\begin{aligned} [\![\Gamma \vdash \lambda x.\, M'' : A]\!]^{\mathcal{M},g}_{\sigma}(b) &= [\![\Gamma, x : B \vdash M'' : C]\!]^{\mathcal{M},g[x\mapsto b]}_{\sigma} && \text{by cond. 4 of def. 34} \\ &= [\![\Gamma, x : B \vdash M'' : C]\!]^{\mathcal{M},g[x\mapsto b]}_{\sigma'} && \text{by IH on } \pi' \\ &= [\![\Gamma \vdash \lambda x.\, M'' : A]\!]^{\mathcal{M},g}_{\sigma'}(b) && \text{by cond. 4 of def. 34} \end{aligned} \qquad (I.17)$$

Hence $(*)$ by functionality.

Case App. As M' is β-normal, we can decompose it as $M' = M'' N_1 ... N_n$ with $M'' = x$ or $M'' = c$ and

$$[\![\Gamma \Vdash M : A]\!]^{\mathcal{M},g}_{\sigma} = [\![\Gamma \Vdash M'' : B_1 \to ... \to B_n \to A]\!]^{\mathcal{M},g}_{\sigma}([\![\Gamma \Vdash N_1 : B_1]\!]^{\mathcal{M},g}_{\sigma})...([\![\Gamma \Vdash N_n : B_n]\!]^{\mathcal{M},g}_{\sigma})$$

If $M'' = c$, then, by calling $C = B_1 \to ... \to B_n \to A$, $C\sigma = \mathtt{t}(c) = C\sigma'$. If $M'' = x$, then $x : C\sigma \in \mathsf{dom}\, g$ and $x : C\sigma' \in \mathsf{dom}\, g$, so $C\sigma = C\sigma'$ by hypothesis on g. In both cases we conclude that for any $i \leq n$, $B_i\sigma = B_i\sigma'$. Therefore we can apply the induction hypothesis on the subproof of $\Gamma \Vdash M'' : C$ and every $\Gamma \Vdash N_i : B_i$. This yields $[\![\Gamma \Vdash M : A]\!]^{\mathcal{M},g}_{\sigma} = [\![\Gamma \Vdash M : A]\!]^{\mathcal{M},g}_{\sigma'}$

Case Exch. Applying the induction hypothesis straightforwardly works.

□

Proposition 7. *The denotation of a derivation only depends on its root typing judgment.*

Proof. Set $\Gamma \vdash M : A$ a valid typing judgment and π_1 and π_2 two proofs of it. By proposition 3, $\Gamma \vdash M : A$ admits a principal derivation π.

As π_1 is a typing derivation, we have $[\![\pi_1]\!]^{\mathcal{M},g} = [\![\pi_1]\!]^{\mathcal{M},g}_{\emptyset}$, where $\emptyset$ stands for the global substitution of empty definition domain. And similarly $[\![\pi_2]\!]^{\mathcal{M},g} = [\![\pi]\!]^{\mathcal{M},g}_{\sigma'}$.

Thus, we can apply lemma 3 and get $[\![\pi_1]\!]^{\mathcal{M},g} = [\![\pi_2]\!]^{\mathcal{M},g}$. □

Remark 10. *According to proposition 1, if M is closed we may write $[\![M]\!]^{\mathcal{M}}$ instead of $[\![\vdash M]\!]^{\mathcal{M},\emptyset}$ to gain clarity. We might also drop the variable type in typing declarations inside typing contexts.*

Relations on class of models

Here we define some key notions about entailment and logical equivalence we will use extensively throughout this thesis.

Definition 36 (Class of models)**.** *A class of models* $\mathbf{k}$ *is a subset of extensional Henkin models on a given signature.*

We make the assumption that in every class of models, every base domain D_a (a atomic) is a poset $(D_a, \leq_a)$. For most atomic types, $\leq_a$ is taken to be the equality on D_a.

Definition 37 (Order on domains)**.** *Set $f, g \in D_{B\to C}$. We define the order $\leq_{B\to C}$ on $D_{B\to C}$ by induction:*

$$f \leq_{B\to C} g \text{ if for all } b \in D_B, f(b) \leq_C g(b) \tag{I.18}$$

Definition 38 (Entailment)**.** *Set a class of models $\mathbf{k}$ on Σ, $\Gamma \vdash_\Sigma M : A$ and $\Delta \vdash_\Sigma N : A$. We say that M **entails** N and we write $M \models^A_{\mathbf{k}} N$ if for all $\mathbf{k}$-model $\mathcal{M}$ and assignment g,*

$$[\![\Gamma \vdash M : A]\!]^{\mathcal{M},g} \leq_A [\![\Delta \vdash N : A]\!]^{\mathcal{M},g}$$

Definition 39 (Logical equivalence)**.** *We define* ***logical equivalence*** *in a class of models* $\mathbf{k}$ *by*

$$M \cong_{\mathbf{k}} N \qquad \text{if } M \models^{A}_{\mathbf{k}} N \text{ and } N \models^{A}_{\mathbf{k}} M \text{ for some } A \tag{I.19}$$

We can prove, that

$$M \cong_{\mathbf{k}} N \qquad \text{iff for every } \mathcal{M} \in \mathbf{k} \text{ and } g, [\![\Gamma \vdash M : A]\!]^{\mathcal{M},g} = [\![\Delta \vdash N : A]\!]^{\mathcal{M},g} \tag{I.20}$$

Even if $\beta\eta$-equivalence is included in logical equivalence, we aim at only using that symbol to refer to special properties of class $\mathbf{k}$*.*

Definition 40 (Upward monotonicity)**.** *Set $\Gamma \vdash_{\Sigma} M : A$. We say that M is* ***upward monotonic*** *in $z : B \in \Gamma$ if for all* $\mathbf{k}$*-model $\mathcal{M}$ and g,*

$$\text{for all } b, b' \in D_B \text{ such that } b \leq_B b' \text{ we have } [\![\Gamma \vdash M : A]\!]^{\mathcal{M},g[z\mapsto b]} \leq_A [\![\Gamma \vdash M : A]\!]^{\mathcal{M},g[z\mapsto b']} \tag{I.21}$$

We say that M is upward monotonic in its argument if $A = B \to C$ and $\Gamma, z : B \vdash M\, z : C$ is upward monotonic in the fresh variable $z : B$.

Proposition 8. *Suppose $\Gamma \vdash_{\Sigma} M : B \to C$ is upward monotonic in its argument.*
If $N \models^{B}_{\mathbf{k}} N'$, then $M\, N \models^{C}_{\mathbf{k}} M\, N'$.

Proof. Suppose $N \models^{B}_{\mathbf{k}} N'$. Set $\mathcal{M}$ a $\mathbf{k}$-model and g an assignment. Applying the definition of the upward monotonicity of M in its argument to $[\![N]\!]^{\mathcal{M},g} \leq_B [\![N']\!]^{\mathcal{M},g}$ gives us

$$[\![\Gamma, z : B \vdash M\, z : C]\!]^{\mathcal{M},g[z\mapsto [\![N]\!]^{\mathcal{M},g}]} \leq_C [\![\Gamma, z : B \vdash M\, z : C]\!]^{\mathcal{M},g[z\mapsto [\![N']\!]^{\mathcal{M},g}]}$$

which implies, as z is a fresh variable

$$[\![\Gamma \vdash M : B \to C]\!]^{\mathcal{M},g}([\![z : B \vdash z : B]\!]^{\mathcal{M},g[z\mapsto [\![N]\!]^{\mathcal{M},g}]}) \leq_C [\![\Gamma \vdash M : B \to C]\!]^{\mathcal{M},g}([\![z : B \vdash z : B]\!]^{\mathcal{M},g[z\mapsto [\![N']\!]^{\mathcal{M},g}]})$$

which is nothing else but

$$[\![\Gamma \vdash M : B \to C]\!]^{\mathcal{M},g}([\![\Delta \vdash N : B]\!]^{\mathcal{M},g}) \leq_C [\![\Gamma \vdash M : B \to C]\!]^{\mathcal{M},g}([\![\Delta \vdash N' : B]\!]^{\mathcal{M},g})$$

So

$$[\![\Gamma, \Delta \vdash M\, N : C]\!]^{\mathcal{M},g} \leq_C [\![\Gamma, \Delta \vdash M\, N' : C]\!]^{\mathcal{M},g}$$

We proved that $M\, N \models^{C}_{\mathbf{k}} M\, N'$. □

II.
Syntax of French questions with ACGs

II.1 Syntax of French questions

We use contemporary French to illustrate our study. We mostly take simple examples, so that the behavior of French speakers regarding them should be univocal. If not specified explicitly, standard French is assumed. For convenience, we will mainly cover written sentences, as string of units called words.

II.1.1 Long-distance dependencies

Relative clauses and questions exhibit some long-distance dependencies in a lot of languages. Let us talk about relative clauses first.

A relative clause is an embedded clause that modifies noun. A relative clause is an adjunct and they stand alone as a sentence. Relative clauses are introduced by a pronoun or pro-expression (e.g. (II-1)). Here we semantically coindex anaphors with their referents.

(1) a. Marie aime le [dessin$_i$ [que$_i$ [j'ai fait hier]]].
b. Marie aime la [région$_i$ [où$_i$ [elle a grandi]]].
c. Marie aime une [personne$_i$ [[pour laquelle$_i$][elle ferait tout]]].

It turns out that there is a systematic way of creating a relative clause from a declarative sentence: we replace the element we want to relativize and put it in first position of the clause. In transformational grammars [10] , this operations is viewed as an **extraction** (wh-movement). We put the coindexed symbol t_i (for *trace*) to indicate the initial place of the moved wh-phrase indexed by i. This relationship between a vacant place and a word can take place through different intermediary clauses, like in (II-2-a).

(2) a. C'est le peintre [à qui$_i$ Marie$_j$ sait [qu'elle$_j$ achetera un tableau t_i].

Questions with an interrogative pronoun (wh-questions) behave similarly.

(3) a. Que$_i$ [veut cet homme t_i] ?
b. Qui$_i$ [penses-tu [qu'il$_j$ invitera t_i]] ?

However, some extractions are not allowed. We call phrases inside which it is not possible to extract strict subconstituents **extraction islands**. Relative clauses and subjects, among others, are considered extraction islands, as shown in (II-4).

(4) a. Jean connait l'homme$_i$ [qui$_i$ [t_i a vu Marie]].
b. *C'est la femme$_j$ [que$_j$ Jean connait l'homme$_i$ [qui$_i$ [t_i a vue t_j]]].
c. Le chien de Marie adore le chat de Jean.
d. De qui$_i$ le chien de Marie adore(-t-il) le chat t_i ?
e. *De qui$_i$ le chien t_i adore le chat de Jean ?

Currently, there is not consensus in the literature whether all island constraints are strong syntactic features or whether they are byproducts of mental process complexity [5].

Note that other syntax models do not treat long-distance dependencies as extractions, like HPSGs [51].

II.1.2 French interrogatives

In French, there are 4 main ways to build an interrogative sentence [31].

The first one consists in inverting the subject and the object, e.g. (II-5-a). This also happens with an interrogative pronoun, e.g. (II-5-b). However, contrary to English, this inversion is not systematic. When the subject of a yes/no question is not a clitic pronoun (je, tu, il, elle, on, nous, vous, ils, elles), a pleonastic clitic pronoun must be added after the verb to maintain the interrogation, like in (II-5-c). This formation is called *retrograde versational interrogation* [20, §.1390].

(5) a. Vient-il ?
b. D'où vient Jean ?
c. Jean vient-il ?
d. *Vient Jean ?

The second method uses the interrogative particle est-ce que (literally meaning "is it that") at the beginning of the sentence, e.g. (II-6-a). If there is an interrogative pronoun, it is placed before est-ce que, e.g. (II-6-b). This marker changes to est-ce qui when the subject is extracted, e.g. (II-6-c).

(6) a. Est-ce que Jean vient ?
b. D'où est-ce que Jean vient ?
c. Qui$_i$ est-ce qui [t_i vient] ?

In colloquial and regional French, est-ce que has variants, pronouncing only est-ce (II-7-a), que (II-7-b) or nothing at all (sometimes present with a right dislocation, like in (II-7-c)) [31].

(7) a. Qu'est-ce tu fais ?
b. Où qu'il va ?
c. Où il$_i$ va, çui$_i$-là ?

French also allows declarative questions, i.e. where only final intonation indicates interrogation. Raising (resp. falling) intonation is written with an up arrow (resp. down arrow), e.g. (II-8-a). Contrary to English, where such question types are only used to ask for checking (e.g. by misunderstanding), they are quite widespread in colloquial French. Complex *NP* might be dislocated on the left, and disjunctive help ou pas might be added at the end, see (II-8-b).

This construction is very similar to a total absence of est-ce que in an est-ce que question. However, contrary to the latter, interrogative pronouns may stay *in situ*, i.e. at their original place according to transformational grammars [9], e.g. (II-8-c).

(8) a. Il vient-↑?
b. Il$_i$ vient-↑, Jean$_i$, ou pas-↓?
c. Il vient d'où-↑?

Generalized uses of $...t-il$ (and euphonic t for verbs of the first group, e.g. Que mange-t-il ?) combined with a muting of the final l lead to a fixed interrogative particle -ti, like in (II-9-a). In Québecois French, this particle evolved into -tu [58], and is now widespread in colloquial speech, e.g. (II-9-b), (II-9-c). However, it is only possible to create yes/no questions with -tu, and this particle does not behave properly with negation [50].

(9) a. C'est-ti pas fini ?
b. Ah, on mange-tu un petit peu ?
c. T'as-tu vu mon parapluie ?

II.2 Abstract categorial grammars

Abstract categorial grammars (ACGs) are built on simply-typed λ-calculus. After the formal definitions in section II.2.1, we remind main properties of ACGs: order hierarchy in II.2.2, relationship with context-free formalisms in II.2.3, common additional extensions in II.2.4 and complexity results in II.2.5.

II.2.1 Definitions

Definition 41 (Morphism of HSO)**.** *A (resp. linear) morphism of higher-order signatures* $\mathfrak{L} : \Sigma_1 \to \Sigma_2$ *is a couple* $\mathfrak{L} = (F, G)$ *(*$\Sigma_i = (\mathcal{B}_i, \mathcal{C}_i, \mathtt{t}_i)$*) such that*

- $F : \mathcal{B}_1 \to \mathcal{T}(\mathcal{B}_2)$ *(resp.* $F : \mathcal{B}_1 \to \mathcal{T}^\circ(\mathcal{B}_2)$*)*
- $G : \mathcal{C}_1 \to \Lambda(\Sigma_2)$ *(resp.* $G : \mathcal{C}_1 \to \Lambda^\circ(\Sigma_2)$*)*
- *for all* $c \in \mathcal{C}_1$, $\vdash_{\Sigma_2} G(c) : F(\mathtt{t}_1(c))$

F extends to a morphism from $\mathcal{T}(\mathcal{B}_1)$ *to* $\mathcal{T}(\mathcal{B}_2)$ *by induction:* $F(A \to B) = F(A) \to F(B)$.

G extends to a morphism from $\Lambda(\Sigma_1)$ *to* $\Lambda(\Sigma_2)$*. To compute* $G(M)$*, assume* $M \in \Lambda(\Sigma_1)$ *is* α*-renamed to ensure: for every constant* $c \in \mathcal{C}_1$ *occurring in* M*, we have* $(\mathsf{FV}(M) \cup \mathsf{BV}(M)) \cap \mathsf{FV}(G(c)) = \emptyset$*. Then we can define:*

$$G(x) = x \qquad G(M'\,N) = G(M')\,G(N) \qquad G(\lambda x.\,N) = \lambda x.\,G(N) \tag{II.1}$$

By abuse of notation, we write $\mathfrak{L}$ *instead of* F *or* G *or their extensions.*

Definition 42 (ACG)**.** *An* ***abstract categorial grammar (ACG)*** *is a tuple* $\mathcal{G} = (\Sigma_1, \Sigma_2, \mathfrak{L}, S)$*, where* $\mathfrak{L}$ *is a morphism between the higher-order signatures* Σ_1 *and* Σ_2*, and* S *is a distinguished type of* Σ_1*.*

Higher-order signatures are also called ***vocabularies*** *and morphisms are called* ***lexicons****. If* $\mathfrak{L}_1^2 : \Sigma_1 \to \Sigma_2$*, we usually call* Σ_1 *the* ***abstract vocabulary*** *and* Σ_2 *the* ***object vocabulary****.*

Remark 11. *Traditional ACGs are linear lexicons between linear signatures. The properties stated in sections II.2.2, II.2.3 and II.2.5 only concern linear ACGs.*

Definition 43 (Lexicalized ACG)**.** *A lexicon* $\mathfrak{L}$ *is lexicalized if for every abstract constant* $c_1 \in \mathcal{C}_1$*,* $\mathfrak{L}(c_1)$ *contains at least one object constant of* $\mathcal{C}_2$*.*

Definition 44 (Languages)**.** *The abstract language* $\mathcal{A}(\mathcal{G})$ *of an ACG* $\mathcal{G} = (\Sigma_1, \Sigma_2, \mathfrak{L}, S)$ *is* $\Lambda(\Sigma_1)^S_\emptyset$*, the set of closed* λ*-terms of type* S*.*

The object language $\mathcal{O}(\mathcal{G})$ *of* $\mathcal{G}$ *is* $\mathfrak{L}(\Lambda(\Sigma_1)^S_\emptyset)$*. We usually consider* λ*-terms up to* $\beta\eta$*-equivalence.*

The traditional ACG framework is the one of Fig. II.1 . A deep structure λ-term M can be mapped to its surface structure $\mathcal{Y}(M)$ (its yield) or its semantic representation $\mathfrak{L}_{\mathbf{e}}(M)$. The distinguished type of $\Sigma_{\mathbf{D}}$ is usually s (sentence), $\mathcal{Y}(\mathsf{s}) = \mathsf{str}$ (string) $\mathfrak{L}_{\mathbf{e}}(\mathsf{s}) = o$ (truth value). The language of this framework is the object language of $\mathcal{Y}$.

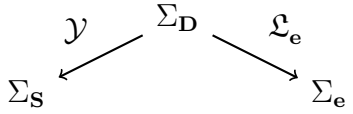

Figure II.1 – Traditional ACG for NLP

In practice, using an ACG for natural language processing (NLP) follows this procedure: Given a surface string w of words belonging to $\Sigma_{\mathbf{S}}$, we parse it, i.e. we find all antecedents M of w by $\mathcal{Y}$. Then, we can compute the semantic interpretations of these M by applying $\mathfrak{L}_{\mathbf{e}}$.

II.2.2 Order hierarchy

Definition 45 (Order of a type)**.** *The* ***order*** *of a simple type* $\mathsf{ord}(A)$ *is defined inductively by:*

$$\mathsf{ord}(a) = 1 \text{ if } a \in \mathcal{B} \qquad \mathsf{ord}(A \to B) = \max(\mathsf{ord}(A) + 1, \mathsf{ord}(B)) \tag{II.2}$$

The order of a type indicates how much "intricate" the objects of that type can be. Constants have order 1, simple functions have order 2, function of functions have order 3, and so on.

Definition 46 (Order of an ACG)**.** *The order of a vocabulary* Σ *is the maximal order of the types* $\mathtt{t}(c)$ *among constants* $c \in \mathcal{C}$*.*

The order of a lexicon $\mathfrak{L} : \Sigma_1 \to \Sigma_2$ *is the maximal order of image types* $\mathfrak{L}(a_1)$ *among abstract atomic types* $a_1 \in \mathcal{C}_1$*.*

The class $\mathbf{G}(m, n)$ *is the set of ACGs which abstract vocabulary is of order at most* m *and which lexicon is of order at most* n*. The class of languages* $\mathbf{L}(m, n)$ *is the set of object languages of* $\mathbf{G}(m, n)$ *ACGs.*

We call m^{th}*-order ACGs the elements of* $\mathbf{G}(m, n)$ *for some* n*.*

Here, we only focus on ACG which object language is made of strings of words (and not on derivation trees), as in section II.3.3.

Theorem 3. *For every* $m, n \geq 1$*,* $\mathbf{L}(m, n+1) \subseteq \mathbf{L}(m+1, n)$ *and* $\mathbf{L}(m+3, n) \subseteq \mathbf{L}(m+2, n)$*.*

Therefore there are only three linear hierarchies of interest : first-order, second-order and third-order ACGs. In practice, we usually consider second-order ACGs because they have interesting complexity properties. Moreover, only second order ACGs can be parsed with ACGtk, the toolkit designed for ACGs [52].

II.2.3 Connection with other formalisms

ACGs are rather general, in the sense that they are able to simulate other models of natural language syntax.

Theorem 4. *Context-free grammars can be encoded in* $\mathbf{G}(2,2)$ *ACGs, i.e. if* $\mathcal{G}$ *is a context-free grammar, there exists a second-order ACG* $\mathcal{G}'$ *such that the object language of* $\mathcal{G}$ *and* $\mathcal{G}'$ *are equal.*

Theorem 5. *Tree-adjoining grammars can be encoded in* $\mathbf{G}(2,3)$ *ACGs.*

The proofs are given in [28] and [22] respectively. Moreover, this equivalence is strong, i.e. the derivations of the created ACG are the same as the one of the original CFG / TAG.

Lambek grammars can also be encoded in ACGs [24]. The translation exposed uses CFGs as intermediary system. It has the drawback to create a lot of redundant constants, which increases the number ambiguities (and thus drops the efficiency) when parsing.

Another translation, given in [54] exploits typing derivations to build a second-order ACG generating the same derivations as a large enough fragment of a given Lambek grammar. This construction is claimed to be sufficient to parse almost all natural language sentences given the universal constraints on extractions ([53], see section II.1.1).

Theorem 6. *Every linear context-free rewriting system can be encoded in a ACG of* $\mathbf{G}(2,4)$ *[28].*
Every second order ACG can be transformed into a linear context-free rewriting system [56].

Corollary 1 (Second order hierarchy collapse)**.** *If* $\mathcal{G}$ *is a second-order ACG, then there exists an ACG* $\mathcal{G}'$ *of* $\mathbf{G}(2,4)$ *such that* $\mathcal{O}(\mathcal{G}) = \mathcal{O}(\mathcal{G}')$*.*

II.2.4 Additional structures on ACGs

The set of traditional categorial types is very coarse and thus fails to capture dependencies inside argument relations. A solution is to add feature structures to types, similarly to HPSGs.

An extension of simple types to dependent products, unit type and disjoint union has been suggested [26]. This solution can simulate feature structures and account for agreement (in person, number, gender, case,...). This construction creates a bridge between ACGs and natural language models using dependent record type theory [18].

These extensions can also be used to add restrictions on derivations, e.g. to model island constraints [53].

II.2.5 Complexity

Natural language processing (NLP) is also interested in the efficiency. This can be measured by the complexity of elementary computation or decision problems.

Definition 47 (Membership problems)**.** *The membership problem of a class* $\mathbf{C}$ *of ACGs is, given an ACG* $\mathcal{G} \in \mathbf{C}$*, the following:*

$$\begin{cases} \text{INPUT}: & M \in \Lambda(\Sigma_2) \\ \text{QUESTION}: & \textit{Do we have } M \in \mathcal{O}(\mathcal{G})\textit{?} \end{cases} \tag{II.3}$$

The universal membership problem of a class $\mathbf{C}$ *of ACGs is the following:*

$$\begin{cases} \text{INPUT}: & \mathcal{G} \in \mathbf{C}, M \in \Lambda(\Sigma_2) \\ \text{QUESTION}: & \textit{Do we have } M \in \mathcal{O}(\mathcal{G})\textit{?} \end{cases} \tag{II.4}$$

Theorem 7. *The membership problem of lexicalized second-order ACGs is polynomial [57].*

Given a second-order ACG, a term of size n can be parsed in time $\mathcal{O}(n^6)$ and in space $\mathcal{O}(n^5)$ [40].

Theorem 8 (Salvati [57])**.** *The universal membership problem of lexicalized $\mathbf{G}(2,2)$ is NP-complete.*

There exists a lexicalized ACG $\mathcal{G}$ of $\mathbf{G}(3,1)$ such that the membership problem on $\mathcal{G}$ is NP-complete.

Theorem 9. *ACGs with cartesian product are Turing-complete [27].*

To sum it up, second-order ACGs are tractable and can be efficiently used to parse natural language sentences. Wide-coverage grammars can be retrieved from other models, such as TAGs. Other methods [45] have also been developed to constitute large digital lexicons for ACGtk from raw text data.

II.3 Syntax modelization

II.3.1 Toy fragment of French

In this section, we analyze the syntax of toy fragment of French. This fragment is constituted to exhibit simple interesting phenomena about intentions and questions. In this thesis, we only study questions built with est-ce que and declarative interrogations (see section II.1.2), because their syntactic behaviors are easier to model. The syntax of other question forms are left for future work. Note that at the semantic level, though, there is no more distinction between these forms.

The first set (II-10) contains our basis of declarative sentences. We select an intransitive verb for simplicity. Transitive verbs and object extraction do not raise other difficulties.

(10) a. Marie dort.
b. Marie dort chez Camille.

The second set (II-11) contains modalities to explore intentional semantics (more in section III.3).

(11) a. Marie dort peut-être.
b. Jean sait que Marie dort.

The third set (II-12) contains declarative interrogatives and *in situ* interrogative pronouns like in (II-12-b) and (II-12-c). Note that (II-12-b) turns out to have the same surface form as a subject retrograde versational interrogative.

(12) a. Marie dort-↑?
b. Qui-↑ dort ?
c. Marie dort où-↑?

The fourth set (II-13) contains questions with the particle est-ce que.

(13) a. Est-ce que Marie dort ?
b. Qui est-ce qui dort ?
c. Où est-ce que Marie dort ?

Finally, set (II-14) contains est-ce que interrogative clauses embedded under a *responsive veridical* verb [43].

(14) a. Jean sait si Marie dort.
b. Jean sait qui est-ce qui dort.
c. Jean sait où est-ce que Marie dort.

II.3.2 Deep syntax

Following the tradition of categorial grammars, we use the syntactic type s for a (possibly embedded) clause, np for a noun phrase and n for a noun. To differentiate declarative clauses from interrogative ones, we add a type $\mathsf{s}^?$.

As both declarative and interrogative clauses may be full sentences, we add another type s^p into which s and $\mathsf{s}^?$ can be transformed. We analyze written forms here, so these transformations can be lexicalized by the final period or interrogation mark, so that s^p stands for punctuated sentence. Similarly, we distinguish declarative and interrogative complementizer phrases cp and $\mathsf{cp}^?$.

An ontology of types could better capture syntactic phenomena. However, the focus of this work is not on syntax, so we leave this for future work. We also do not provide feature structures to account for agreement.

The basic types of the deep syntax linear signature $\Sigma_{\mathbf{D}}$ are $\mathcal{B}_{\mathbf{D}} \mathrel{\hat{=}} \{\mathsf{s}^p, \mathsf{s}, \mathsf{s}^?, \mathsf{np}, \mathsf{n}, \mathsf{cp}, \mathsf{cp}^?\}$. We could have also added pp for argument prepositional phrases, but we only consider adjunct PP here. The table of types constants of $\Sigma_{\mathbf{D}}$ is displayed in Tab. II.1.

DECL :	$\mathsf{s} \multimap \mathsf{s}^p$	DORT :	$\mathsf{np} \multimap \mathsf{s}$
INT :	$\mathsf{s}^? \multimap \mathsf{s}^p$	SAIT_1 :	$\mathsf{cp} \multimap \mathsf{np} \multimap \mathsf{s}$
DECL-INT :	$\mathsf{s}^? \multimap \mathsf{s}^p$	SAIT_2 :	$\mathsf{cp}^? \multimap \mathsf{np} \multimap \mathsf{s}$
PEUT-ÊTRE :	$\mathsf{s} \multimap \mathsf{s}$	CHEZ :	$\mathsf{np} \multimap (\mathsf{np} \multimap \mathsf{s}) \multimap (\mathsf{np} \multimap \mathsf{s})$
proper name :	np	QUE :	$\mathsf{s} \multimap \mathsf{cp}$
SI :	$\mathsf{s} \multimap \mathsf{cp}^?$	ECQ_1 :	$\mathsf{s} \multimap \mathsf{s}^?$
QUI :	$(\mathsf{np} \multimap \mathsf{s}) \multimap \mathsf{s}^?$	$\text{ECQ}_{2.1}$:	$(\mathsf{np} \multimap \mathsf{s}) \multimap ((\mathsf{np} \multimap \mathsf{s}) \multimap \mathsf{s}^?) \multimap \mathsf{s}^?$
		$\text{ECQ}_{2.2}$:	$(\mathsf{np} \multimap \mathsf{s}) \multimap ((\mathsf{np} \multimap \mathsf{s}) \multimap \mathsf{s}^?) \multimap \mathsf{cp}^?$
OÙ :	$A_{\mathbf{D}}$	$\text{ECQ}_{3.1}$:	$(A_{\mathbf{D}} \multimap \mathsf{s}^?) \multimap A_{\mathbf{D}} \multimap \mathsf{s}^?$
where $A_{\mathbf{D}} \mathrel{\hat{=}}$	$(\mathsf{np} \multimap \mathsf{s}) \multimap (\mathsf{np} \multimap \mathsf{s}^?)$	$\text{ECQ}_{3.2}$:	$(A_{\mathbf{D}} \multimap \mathsf{s}^?) \multimap A_{\mathbf{D}} \multimap \mathsf{cp}^?$

Table II.1 – Constants of the deep syntax vocabulary $\Sigma_{\mathbf{D}}$

ECQ stands for EST-CE QUE. Versions 1 form yes/no questions, versions 2.x form subject questions and version 3.x form adjunct questions. PEUT-ÊTRE is taken here as a clause modifier for simplicity, and not as a VP modifier, like OÙ or CHEZ CAMILLE. SAIT_1 takes declarative complementizer phrases as complement, whereas SAIT_2 taken interrogative ones.

Distinguishing complementizer phrases from clause types allows to reject sentences like *Que Marie dort. or *Si Marie dort ?. However, this has the drawback to ask clauses with no complementizer to still generate a cp or $\mathsf{cp}^?$ type to be able to be embedded. The transformational approach, i.e. a silent interrogative complementizer $\mathsf{s}^? \multimap \mathsf{cp}^?$, would produce a non-lexicalized term, what is unwanted in ACGs. Not only does this increase parsing difficulty, but this would also allow declarative interrogative clauses to be embedded, like *Je sais Marie dort où.

Similarly, setting ECQ's to have $\mathsf{cp}^?$ as target type *only* plus adding non-lexicalized $\mathsf{cp}^? \multimap \mathsf{s}^?$ operator to allow stand-alone est-ce que questions would lose the interest of separating $\mathsf{s}^?$ from $\mathsf{cp}^?$, as stated above.

The solution we propose is to split ECQ's into one kind (x.1) producing $\mathsf{s}^?$ to stand alone, and one other (x.2) producing $\mathsf{cp}^?$ to be embedded. This may not be elegant and optimal. Nevertheless,

this split has the advantage to be lexically specifiable. In particular, ECQ_1 should not be split, in order to reject Je sais est-ce que Marie dort in aid of (II-14-a).

The parse terms of set 1 are:

(15) a. $\textsc{decl}\,(\textsc{dort}\;\textsc{Marie})$
b. $\textsc{decl}\,(\textsc{chez}\;\textsc{Camille}\;\textsc{dort}\;\textsc{Marie})$

The parse terms of set 2 are:

(16) a. $\textsc{decl}\,(\textsc{peut-être}\,(\textsc{dort}\;\textsc{Marie}))$
b. $\textsc{decl}\,(\textsc{sait}_1\,(\textsc{que}\,(\textsc{dort}\;\textsc{Marie}))\;\textsc{Jean})$

The parse terms of declarative questions are:

(17) a. $\textsc{decl-int}\,(\textsc{dort}\;\textsc{Marie})$
b. $\textsc{int}\,(\textsc{qui}\;\textsc{dort})$
c. $\textsc{int}\,(\textsc{où}\;\textsc{dort}\;\textsc{Marie})$

Only polar declarative questions use DECL-INT. Reasoning with phonological forms, this analysis would still be relevant. Indeed, DECL-INT would carry a intonation morpheme of tone raising as lexical anchor (corresponding to the interrogation mark), as suggested by [9]. The tone raising morpheme would already be part of the lexical anchors for constants OÙ and QUI.

The parse terms of est-ce que questions are:

(18) a. $\textsc{int}\,(\text{ECQ}_1\,(\textsc{dort}\;\textsc{Marie}))$
b. $\textsc{int}\,(\text{ECQ}_{2.1}\;\textsc{dort}\;\textsc{qui})$
c. $\textsc{int}\,(\text{ECQ}_{3.1}\,(\lambda^\circ f^{A_\mathbf{D}}.\,f\;\textsc{dort}\;\textsc{Marie})\;\textsc{où})$

The parse terms of embedded questions are:

(19) a. $\textsc{decl}\,(\textsc{sait}_2\,(\textsc{si}\,(\textsc{dort}\;\textsc{Marie}))\;\textsc{Jean})$
b. $\textsc{decl}\,(\textsc{sait}_2\,(\text{ECQ}_{2.2}\;\textsc{dort}\;\textsc{qui})\;\textsc{Jean})$
c. $\textsc{decl}\,(\textsc{sait}_2\,(\text{ECQ}_{3.2}\,(\lambda^\circ f^{A_\mathbf{D}}.\,f\;\textsc{dort}\;\textsc{Marie})\;\textsc{où})\;\textsc{Jean})$

Our analysis is far from perfect. The absence of directionality in types are practical to treat verb objects similarly to subjects. Unfortunately, this behavior is sometimes problematic. Any constant selecting a *VP* type $\mathsf{np} \multimap \mathsf{s}$ is bound to also allow non-continuous constituents "subject + transitive verb lacking a direct object", as [49] explains. This can cause overgeneration.

To see this, take the transitive verb $\textsc{voit} : \mathsf{np} \multimap \mathsf{np} \multimap \mathsf{s}$. We can make Jean voit Camille (II-20-a) and question the object: Jean voit qui ? (II-20-b). However, we have no way to force $\text{ECQ}_{2.1}$ to select the subject, which let the sentence *Qui est-ce qui Jean voit ? (II-20-c) be derivable (see $\mathfrak{L}_\mathbf{S}(\text{ECQ}_{2.1})$ in Tab. II.2).

(20) a. $\textsc{decl}\,(\textsc{voit}\;\textsc{Camille}\;\textsc{Jean})$
b. $\textsc{decl-int}\,(\textsc{qui}\,(\lambda^\circ x^{\mathsf{np}}.\,\textsc{voit}\;x\;\textsc{Jean}))$
c. $\textsc{int}\,(\text{ECQ}_{2.1}\,(\lambda^\circ x^{\mathsf{np}}.\,\textsc{voit}\;x\;\textsc{Jean})\;\textsc{qui})$

The most simple way of recovering grammatical functions is by refining types. Having a subject *NP* type $\mathsf{np}_{\mathsf{subj}}$ and an object one $\mathsf{np}_{\mathsf{obj}}$, we could solve the issue by taking $\textsc{voit} : \mathsf{np}_{\mathsf{obj}} \multimap \mathsf{np}_{\mathsf{subj}} \multimap \mathsf{s}$. We do not implement this for the sake of legibility.

II.3.3 Surface syntax

The surface syntax linear signature $\Sigma_{\mathbf{S}}$ is based on strings. The easy way to define strings in λ-calculus is to let $\mathcal{B}_{\mathbf{S}} \mathrel{\hat{=}} \{*\}$ and define $\mathsf{str} \mathrel{\hat{=}} * \multimap *$. Given a variable $x : *$, the composition of functions $f, g : \mathsf{str}$ as $f\,(g\,x)$ is interpreted as the sequence of symbols $f\ g$.

We define the identity on a type $A \in \mathcal{T}(\mathcal{B}_{\mathbf{S}})$ as $\mathsf{id}_A \mathrel{\hat{=}} \lambda x^A.\, x$. The empty string is $\varepsilon \mathrel{\hat{=}} \mathsf{id}_* : \mathsf{str}$. The concatenation operation can be defined by $+ \mathrel{\hat{=}} \lambda^\circ f^{\mathsf{str}}, g^{\mathsf{str}}, x^*.\, f\,(g\,x)$. Thanks to confluence and strong normalization of β-reduction, we know that $+$ (written as an infix operator) is associative, i.e. $f + (g + h) =_\beta (f + g) + h =_\beta \lambda^\circ x^*.\, f\,(g\,(h\,x))$.

The yield lexicon $\mathcal{Y} : \Sigma_{\mathbf{D}} \to \Sigma_{\mathbf{S}}$ maps every basic type to str. For every lexical item concerned, we create a corresponding constant of type str in $\Sigma_{\mathbf{S}}$, e.g. $\mathsf{dort} : \mathsf{str}$. The surface interpretation of constants is given in Tab. II.2. For an transitive verb, we would have $\mathcal{Y}(\textsc{voit}) = \lambda^\circ y^{\mathsf{str}}, x^{\mathsf{str}}.\, x + \mathsf{voit} + y$.

$$
\begin{array}{rlrl}
\mathcal{Y}(\textsc{decl}) \mathrel{\hat{=}} & \lambda^\circ x^{\mathsf{str}}.\, x + . & \mathcal{Y}(\textsc{int}) = \mathcal{Y}(\textsc{decl-int}) \mathrel{\hat{=}} & \lambda^\circ x^{\mathsf{str}}.\, x + ? \\
\mathcal{Y}(\textsc{dort}) \mathrel{\hat{=}} & \lambda^\circ x^{\mathsf{str}}.\, x + \mathsf{dort} & \mathcal{Y}(\textsc{sait}_1) = \mathcal{Y}(\textsc{sait}_2) \mathrel{\hat{=}} & \lambda^\circ y^{\mathsf{str}}, x^{\mathsf{str}}.\, x + \mathsf{sait} + y \\
\mathcal{Y}(\textsc{peut-être}) \mathrel{\hat{=}} & \lambda^\circ x^{\mathsf{str}}.\, x + \mathsf{peut\text{-}être} & \mathcal{Y}(\textsc{chez}) \mathrel{\hat{=}} & \lambda^\circ y^{\mathsf{str}}, f^{\mathsf{strstr}}, x^{\mathsf{str}}.\, (f\,x) + \mathsf{chez} + y \\
\mathcal{Y}(\langle \textsc{name} \rangle) \mathrel{\hat{=}} & \langle \mathsf{name} \rangle & \mathcal{Y}(\textsc{que}) \mathrel{\hat{=}} & \lambda^\circ x^{\mathsf{str}}.\, \mathsf{que} + x \\
\mathcal{Y}(\textsc{si}) \mathrel{\hat{=}} & \lambda^\circ x^{\mathsf{str}}.\, \mathsf{si} + x & \mathcal{Y}(\mathrm{ECQ}_1) \mathrel{\hat{=}} & \lambda^\circ x^{\mathsf{str}}.\, \mathsf{est\text{-}ce\ que} + x \\
\mathcal{Y}(\textsc{qui}) \mathrel{\hat{=}} & \lambda^\circ f^{\mathsf{strstr}}.\, f\,(\mathsf{qui}) & \mathcal{Y}(\textsc{où}) \mathrel{\hat{=}} & \lambda^\circ f^{\mathsf{strstr}}, x^{\mathsf{str}}.\, (f\,x) + \mathsf{où} \\
\multicolumn{4}{c}{\mathcal{Y}(\mathrm{ECQ}_{2.1}) = \mathcal{Y}(\mathrm{ECQ}_{2.2}) \mathrel{\hat{=}} \lambda^\circ f^{\mathsf{strstr}}, g^{(\mathsf{strstr})\mathsf{str}}.\, (g\,\mathsf{id}_{\mathsf{str}}) + \mathsf{est\text{-}ce\ qui} + (f\,\varepsilon)} \\
\multicolumn{4}{c}{\mathcal{Y}(\mathrm{ECQ}_{3.1}) = \mathcal{Y}(\mathrm{ECQ}_{3.2}) \mathrel{\hat{=}} \lambda^\circ g^{A_{\mathbf{S}}\mathsf{str}}, h^{A_{\mathbf{S}}}.\, (h\,\mathsf{id}_{\mathsf{str}}\,\varepsilon) + \mathsf{est\text{-}ce\ que} + (g\,(\lambda^\circ k^{\mathsf{strstr}}, y^{\mathsf{str}}.\, k\,y))} \\
\multicolumn{4}{c}{\text{where } A_{\mathbf{S}} \mathrel{\hat{=}} (\mathsf{str} \multimap \mathsf{str}) \multimap \mathsf{str} \multimap \mathsf{str}}
\end{array}
$$

Table II.2 – Yield lexicon $\mathcal{Y}$ from $\Sigma_{\mathbf{D}}$ to $\Sigma_{\mathbf{S}}$

Technically, there is no space between final period and the last word of a French sentence according to the norm. However, we suppose for parsing that every sentence is normalized and tokenized, namely words are lowercased, est-ce que is regrouped in a single unit and the final period as a separate lexical unit.

We can check that all surface interpretations β-reduce to the target sentences of section II.3.1. We provide here some of the longest derivations.

Here is derivation of (II-10-b) from interpreting (II-15-b).

$$
\begin{array}{l}
\mathcal{Y}(\textsc{decl}\,(\textsc{chez}\,\textsc{Camille}\,\textsc{dort}\,\textsc{Marie})) \\
\quad = \mathcal{Y}(\textsc{decl})\,(\mathcal{Y}(\textsc{chez})\,\mathcal{Y}(\textsc{Camille})\,\mathcal{Y}(\textsc{dort})\,\mathcal{Y}(\textsc{Marie})) \\
\quad = (\lambda^\circ y^{\mathsf{str}}.\, y + .)\,(\mathcal{Y}(\textsc{chez})\,\mathsf{Camille}\,(\lambda^\circ x^{\mathsf{str}}.\, x + \mathsf{dort})\,\mathsf{Marie}) \\
\quad \to_\beta (\mathcal{Y}(\textsc{chez})\,\mathsf{Camille}\,(\lambda^\circ x^{\mathsf{str}}.\, x + \mathsf{dort})\,\mathsf{Marie}) + . \\
\quad = ((\lambda^\circ y^{\mathsf{str}}, f^{\mathsf{strstr}}, z^{\mathsf{str}}.\, (f\,z) + \mathsf{chez} + y)\,\mathsf{Camille}\,(\lambda^\circ x^{\mathsf{str}}.\, x + \mathsf{dort})\,\mathsf{Marie}) + . \qquad \text{(II.5)} \\
\quad \to_\beta ((\lambda^\circ f^{\mathsf{strstr}}, z^{\mathsf{str}}.\, (f\,z) + \mathsf{chez} + \mathsf{Camille})\,(\lambda^\circ x^{\mathsf{str}}.\, x + \mathsf{dort})\,\mathsf{Marie}) + . \\
\quad \to_\beta ((\lambda^\circ z^{\mathsf{str}}.\, ((\lambda^\circ x^{\mathsf{str}}.\, x + \mathsf{dort})\,z) + \mathsf{chez} + \mathsf{Camille})\,\mathsf{Marie}) + . \\
\quad \to_\beta ((\lambda^\circ x^{\mathsf{str}}.\, x + \mathsf{dort})\,\mathsf{Marie} + \mathsf{chez} + \mathsf{Camille}) + . \\
\quad \to_\beta \mathsf{Marie} + \mathsf{dort} + \mathsf{chez} + \mathsf{Camille} + .
\end{array}
$$

Here is derivation of (II-13-c) from interpreting (II-18-c):

$$
\begin{aligned}
&\mathcal{Y}(\textsc{int}\,(\mathrm{ECQ}_{3.1}\,(\lambda^\circ f^{A_{\mathbf{D}}}.\, f\ \textsc{dort}\ \textsc{Marie})\,\textsc{où}))\\
&\quad\to_\beta (\mathcal{Y}(\mathrm{ECQ}_{3.1})\,(\lambda^\circ f^{A_{\mathbf{S}}}.\, f\,\mathcal{Y}(\textsc{dort})\,\textbf{Marie})\,\mathcal{Y}(\textsc{où}))+?\\
&\quad\to^2_\beta (\mathcal{Y}(\textsc{où})\,\mathsf{id}_{\mathsf{str}}\,\varepsilon) + \textbf{est-ce que} + ((\lambda^\circ f^{A_{\mathbf{S}}}.\, f\,\mathcal{Y}(\textsc{dort})\,\textbf{Marie})\,(\lambda^\circ k^{\mathsf{strstr}}, y^{\mathsf{str}}.\, k\,y))+?\\
&\quad\to_\beta (\mathcal{Y}(\textsc{où})\,\mathsf{id}_{\mathsf{str}}\,\varepsilon) + \textbf{est-ce que} + ((\lambda^\circ k^{\mathsf{strstr}}, y^{\mathsf{str}}.\, k\,y)\,\mathcal{Y}(\textsc{dort})\,\textbf{Marie})+? \qquad \text{(II.6)}\\
&\quad\to^2_\beta ((\lambda^\circ f^{\mathsf{strstr}}, x^{\mathsf{str}}.\, (f\,x) + \textbf{où})\,\mathsf{id}_{\mathsf{str}}\,\varepsilon) + \textbf{est-ce que} + (\mathcal{Y}(\textsc{dort})\,\textbf{Marie})+?\\
&\quad\to^3_\beta (\mathsf{id}_{\mathsf{str}}\,\varepsilon + \textbf{où}) + \textbf{est-ce que} + (\textbf{Marie} + \textbf{dort})+?\\
&\quad=_\beta \textbf{où} + \textbf{est-ce que} + \textbf{Marie} + \textbf{dort}+?
\end{aligned}
$$

The ACG $(\Sigma_{\mathbf{D}}, \Sigma_{\mathbf{S}}, \mathsf{s}^p)$ is in $\mathbf{G}(5,2)$. We could reduce it to $\mathbf{G}(4,2)$ by replacing $\mathsf{np} \multimap \mathsf{s}$ (resp. $\mathsf{np} \multimap \mathsf{s}^?$) by a basic type vp (resp. $\mathsf{vp}^?$). This would solve the several issue of symmetry, at the risk of creating another QUI for object extraction. Nevertheless, to get to a $\mathbf{G}(2,2)$ grammar, serious changes to interrogative pronouns and ECQ's should be done.

III.
Interface with intentional semantics

In this part, we design a syntax-semantics interface with abstract categorial grammars. Section III.1 recalls basic Montagovian semantics in a historical and introductory fashion. In section III.2, we build signature $\Sigma_{\mathbf{e}}$ of extensional semantics and its class of Henkin models. Finally, we design an intentional interpretation $\Sigma_{\mathbf{i}}$ out of $\Sigma_{\mathbf{e}}$ in section III.3 by simulating intentional logic with a class of Henkin models.

III.1 Formal logic semantics

Some logical inferences (e.g. syllogisms) were already known in ancient Greece. At the end of the XIX$^{\text{th}}$ century, Frege set the basis of what is now known under the name of *predicate logic* [62]. His motivation was to formalize logical inferences and give a clean framework to mathematical proofs.

III.1.1 Predicate semantics

In logic, we study statements (aka. **propositions**) and we want to know under which circumstances a statement is true. Truth values are true and false. We use variables $p, q, ...$ to denote a proposition. The *excluded middle* law says that if p is not true, then it is false. The abbreviation *iff* stands for *if and only if* and expresses equivalence between statements. The common operations on truth values are

- Conjunction: $p \wedge q$ is true iff p is true and q is true
- Disjunction: $p \wedge q$ is true iff p is true or q is true or both are true
- Negation: $\neg p$ is true iff p is not true
- Implication: $p \to q$ is true iff q is true whenever p is true
- Equivalence: $p \leftrightarrow q$ is true iff p and q are both true, or p and q are both false.
- Tautology: $\top$ (read "top") is always true
- Contradiction: $\bot$ (res. "bottom") is always false

It is common to use the system of Boole [6] and write 1 for true and 0 for false. We write $\mathbb{B} = \{0, 1\}$ the set of 0 and 1, called Booleans. With this, logical operation translate to arithmetic operations, e.g. conjunction become multiplication. In Tab. III.1, example Boolean tables are displayed for disjunction and implication. On the left we can find every combination of values for p and q, one per line, and on the right the result of the Boolean operation. The symbol used to denote a logical operation is called a **connective**.

For example, no matter the value of p and q, $p \to q$ is equivalent to $(\neg p) \vee q$. Similarly, $p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$, and $\neg p \equiv p \to \bot$, for all p and q.

A n-place **predicate P** is a function of n variables which maps its arguments to a truth value. We assume that there is a set D of all **individuals** (aka. entities) we want to talk about. This set

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

p	q	$p \to q$
0	0	1
0	1	1
1	0	0
1	1	1

Table III.1 – Boolean tables for disjunction (on the left) and implication (on the right).

might be infinite. We use variables $x, y, ...$ to refer to elements of D, also called first-order citizens. With the notation of functions: $\mathbf{P} : D^n \to \mathbb{B}$ with $n \in \mathbb{N}$.

For example **blue** is a predicate, and $\mathbf{blue}(x)$ equals true iff x is blue. Equivalently, we can view **blue** as a subset of D (written $\mathbf{blue} \subseteq D$), and $\mathbf{blue}(x)$ iff x belongs to **blue** (written $x \in \mathbf{blue}$). Generally speaking, it is conventional to confuse a subset $A \subseteq E$ with its **characteristic function** $\chi_A : E \to \mathbb{B}$.

Like adjectives, intransitive verbs are one-place predicates. Transitive verbs are two-place predicates, i.e. they have two arguments: the individual given by the subject and the individual given by the object. For example, $\mathbf{love}(x, y)$ is true iff x loves y.

Formulas are a combination of predicates and connectives. They can represent the semantics of a sentence. For example, $\mathbf{cat}(x) \to \mathbf{sleep}(x)$ states that the sentence Chaque chat dort is true iff for all individual a, if a is a cat, then a sleeps.

In 1967, Davidson [21] suggests a model where the meaning of a sentence is the set of conditions under which this sentence is true. His system, now known under the name *truth-conditional* semantics, was inspired by Tarski's semantic theory of truth.

III.1.2 Montagovian semantics

In 1970, Montague [47, 46] established what is now known as Montague grammar (or Montagovian semantics). His insight was to consider natural language (semantics) and formal languages used in mathematical logic or computer programming as a two cases of a similar framework.

He uses classical **first-order logic**, an extension of predicate logic, as basis. In first-order logic (FOL), we allow variables $x, y, ...$ to be bound by a quantifier. We can also have function symbols $f, g, ...$ of some arity which allow us to create *first-order terms* $u, ...$ (see definition 11 of terms in section I.3.1). Now, we consider a n-place predicate $\mathbf{P}$ as a pure symbol.

A **model** $\mathcal{M}$ is a pair (D, I) where D is a set of individuals and I is an interpretation, i.e.

- for any n-place predicate $\mathbf{P}$, $I(\mathbf{P})$ is a subset of the cartesian product D^n
- for any function symbol f of arity n, $I(f)$ is a function from D^n to D

An **assignment** g is a partial function that maps some variables to individuals. By $g[x \mapsto a]$, we mean the function g with the additional condition that x is mapped to $a \in D$. The Greek letters $\varphi, \psi, ...$ stand for FOL formulas. The statement $g \models \varphi$ signifies that, under the assignment g, φ is true.

Symbol $\hat{=}$ is the equality of definition. The semantics of first-order logic is defined as:

- Variable: $I_g(x) \hat{=} g(x)$
- Function symbol: $I_g(f(u_1, ..., u_n)) \hat{=} I(f)(I_g(u_1), ..., I_g(u_n))$

- Predicate: $\mathcal{M}, g \models \mathbf{P}(u_1, ..., u_n)$ if $(I_g(u_1), ...I_g(u_n)) \in I(\mathbf{P})$
- Logical connectives: $\mathcal{M}, g \models \psi \wedge \psi$ if $g \models \psi$ and $g \models \varphi$

  (similar for the other connectives, see previous subsection)
- Universal quantifier: $\mathcal{M}, g \models \forall x.\, \varphi$ if for all $a \in D$, $g[x \mapsto a] \models \varphi$
- Existential quantifier: $\mathcal{M}, g \models \exists x.\, \varphi$ if there exists $a \in D$ such that $g[x \mapsto a] \models \varphi$

A formula φ is *valid* in a model $\mathcal{M}$ if for all assignment g, $\mathcal{M}, g \models \varphi$ holds. We write $\models \varphi$ if φ is valid in every model, in other words, if φ is a tautology. We write $\varphi \models \psi$ if, for every $\mathcal{M}, g$ such that $\mathcal{M}, g \models \varphi$, we also have $\mathcal{M}, g \models \psi$. This defines the relation of **entailment** between formulas.

Example 4. *We have* $\mathbf{P}(x) \wedge (\mathbf{P}(x) \to \mathbf{Q}(x, y)) \models \exists z.\, \mathbf{Q}(z, y)$.

Proof. Set a model $\mathcal{M}$ and an assignment g such that $\mathcal{M}, g \models \mathbf{P}(x) \wedge (\mathbf{P}(x) \to \mathbf{Q}(x, y))$. It means that $\mathcal{M}, g(x) \in I(\mathbf{P})$, and if $g(x) \in I(\mathbf{P})$ then $(g(x), g(y)) \in I(\mathbf{Q})$. Given this, we can deduce that $(g(x), g(y))$ indeed belongs to $I(\mathbf{Q})$. By taking $a = g(x)$, we found an element of $a \in D$ such that $\mathcal{M}, g[z \mapsto a] \models \mathbf{Q}(z, y)$ because $g[z \to a](z) = g(x)$. Therefore, $\mathcal{M}, g \models \exists z.\, \mathbf{Q}(z, y)$. □

This example illustrates the modus ponens law: if φ and $\varphi \to \psi$, then ψ.

The *semantics* of a lexical unit is a formula. We commonly use double brackets to express this, e.g. $[\![\mathsf{aime}]\!] = \mathbf{love}$. The *meaning* of a sentence is the set of models and assignments under which the semantics of this sentence is true.

Montague uses types to categorize elements of the logic. Propositions have type t (truth value) and individuals have type e (entity). Given to types A and B, the type $A \to B$ denotes the set of functions from elements of type A to elements of type B. A one-place predicate $\mathbf{P}$ has type $e \to t$. A two-place predicate $\mathbf{Q}$ has type $e \to (e \to t)$ (just written $e \to e \to t$ or even eet) because we can see it as a function from x to the one-place predicates $\mathbf{Q}(x)$.

Montague also uses λ-calculus to construct other functions (see section I.3 for a formal introduction to λ-calculus). λ-calculus allows us to see the combination of variables, predicates and connectives as the sole rule of application forming λ-terms. If u is a λ-term, then $\lambda x.\, u$ designates the function that maps x to u.

For example, $\lambda x.\, \exists y.\, \mathbf{love}\, x\, y$ is the function of type $e \to t$ that maps an individual x to true iff there exist some individual y such that x loves y. For any predicate $\mathbf{P}$, we also have $\lambda x.\, \mathbf{P}\, x \cong \mathbf{P}$ because they represent the same function of type $e \to t$.

III.2 Object language

In ACGs, the syntax-semantics interface is expressed by a higher-order signature $\mathfrak{L}_{\mathbf{e}}$ between deep syntax $\Sigma_{\mathbf{D}}$ and the non-linear Montague object semantics signature $\Sigma_{\mathbf{e}}$.

The basic types of $\Sigma_{\mathbf{e}}$ are $\mathcal{B}_{\mathbf{e}} \mathrel{\hat{=}} \{\iota, o\}$. o is the type of **propositions** (truth values) and ι is the type of **individuals** (aka. entities).

The type morphism $\mathfrak{L}_{\mathbf{e}}$ is given in Tab. III.2. According to Montague [48], noun phrase are not just interpreted as individuals, but as set of properties on individuals. This *type raising* allows us to properly model quantification phenomena, in particular determiners.

$$\begin{aligned} \mathfrak{L}_{\mathbf{e}}(\mathsf{s}^p) = \mathfrak{L}_{\mathbf{e}}(\mathsf{s}) = \mathfrak{L}_{\mathbf{e}}(\mathsf{s}^?) = \mathfrak{L}_{\mathbf{e}}(\mathsf{cp}) = \mathfrak{L}_{\mathbf{e}}(\mathsf{cp}^?) &\mathrel{\hat{=}} o \\ \mathfrak{L}_{\mathbf{e}}(\mathsf{n}) &\mathrel{\hat{=}} \iota \to o \\ \mathfrak{L}_{\mathbf{e}}(\mathsf{np}) &\mathrel{\hat{=}} (\iota \to o) \to o \end{aligned}$$

Table III.2 – Type interpretation at the syntax-semantics interface

In section III.2.1, we define the class **e** of Henkin models simulating extensional logic in an algebraic fashion. Section III.2.2 is dedicated to the definition of $\mathfrak{L}_{\mathbf{e}}$ on λ-terms.

III.2.1 Logical setting

We use the same set of logical connectives as in section III.1, which is:

$$\begin{array}{lll} \wedge_{\mathbf{e}} : \ o \to o \to o & \to_{\mathbf{e}} : \ o \to o \to o & \top_{\mathbf{e}} : \ o \\ \vee_{\mathbf{e}} : \ o \to o \to o & \neg_{\mathbf{e}} : \ o \to o & \bot_{\mathbf{e}} : \ o \end{array} \tag{III.1}$$

Binary connectives are written as infix operators. Quantifiers are decomposed so that variable binding may be done with a λ-abstraction.

$$\forall^{\iota}_{\mathbf{e}} : (\iota \to o) \to o \qquad \exists^{\iota}_{\mathbf{e}} : (\iota \to o) \to o \tag{III.2}$$

We still use the notation $\forall x^{\iota}.\, \varphi$ (resp. $\exists x^{\iota}.\, \varphi$) as a substitute for $\forall^{\iota}_{\mathbf{e}}\, (\lambda x^{\iota}.\, \varphi)$ (resp. $\exists^{\iota}_{\mathbf{e}}\, (\lambda x^{\iota}.\, \varphi)$).

Instead of axioms on connectives, we express properties of first-order logic by the denotation of these constants in a class of Henkin models. This approach has been proven to be equivalent to an axiomatic setting [36, 2].

See section I.2.2 for an introduction about relations.

Definition 48 (Poset)**.** *A partially ordered set (poset) is a set E equipped with a reflexive, anti-symmetric and transitive relation $\leq$ on E.*

A poset is bounded if there exists a greatest element 1_E (i.e. for all $a \in E, a \leq 1_E$) and a lowest element 0_E (i.e. for all $a \in E, 0_E \leq a$).

Definition 49 (Meet and Join)**.** *Set $(E, \leq)$ a poset and $a, b \in E$.*

The greatest lower bound (or meet) of a subset $X \subseteq E$ is an element $c \in E$ (if it exists) such that

1. *for all $a \in X$, $c \leq a$*
2. *for all d such that for all $a \in X$, $d \leq a$, we have $d \leq c$*

The least upper bound (or join) of a subset $X \subseteq E$ is an element $c \in E$ (if it exists) such that

1. *for all $a \in X$, $a \leq c$*
2. *for all d such that for all $a \in X$, $a \leq d$, we have $c \leq d$*

Proposition 9. *If the meet (resp. join) of a set X exists in E, it is unique and we write it $\bigwedge_E X$ (resp. $\bigvee_E X$), and $a \vee_E b$ (resp. $a \wedge_E b$) for a pair $X = \{a, b\}$. Meets have the following properties*

Commutativity $a \wedge_E b = b \wedge_E a$

Associativity $(a \wedge_E b) \wedge_E c = a \wedge_E (b \wedge_E c)$

and same for joins.

If E is bounded, then we also have $a \wedge_E 1_E = a$, $a \wedge_E 0_E = 0_E$, $a \vee_E 1_E = 1_E$ and $a \vee_E 0_E = a$ for all $a \in E$.

Definition 50 (Lattices)**.** *A lattice $(E, \leq, \wedge_E, \vee_E)$ is a poset where all meets and joins of two-element sets exist.*

A distributive lattice is a lattice such that for all $a, b, c \in E$, we have

$$a \vee_E (b \wedge_E c) = (a \vee_E b) \wedge_E (a \vee_E c) \tag{III.3}$$

A complemented lattice is a lattice where for every $a \in E$, there exists a complement $c \in E$ such that

$$a \wedge_E c = 0_E \qquad \text{and} \qquad a \vee_E c = 1_E \tag{III.4}$$

A complete lattice is a lattice where all meets and joins exist.

Proposition 10. *In a distributive lattice, for all $a, b, c \in E$, we also have*

$$a \wedge_E (b \vee_E c) = (a \wedge_E b) \vee_E (a \wedge_E c) \tag{III.5}$$

If the complement of an element a exist, it is unique and we write it $\neg_E a$.

Definition 51 (Boolean algebra)**.** *A **complete Boolean algebra** $(E, \leq, 0_E, 1_E, \wedge_E, \vee_E, \neg_E, \to_E, \bigwedge_E, \bigvee_E)$ is a bounded complete complemented distributive lattice.*

If $a, b \in E$, the pseudocomplement of a relative to b is defined as $a \to_E b \mathrel{\hat{=}} (\neg_E a) \vee_E b$.

Example 5. *The set of Booleans $\mathbb{B} = \{0, 1\}$ with truth operations defined in section III.1 is a Boolean algebra.*

Example 6. *If E is a set, the set $\wp(E)$ of subsets of E (which is in correspondence with the set of characteristic functions $\chi : E \to \mathbb{B}$) is a Boolean algebra with $0_{\wp(E)} = \emptyset$ (empty set), $1_{\wp(E)} = E$ (whole set), $\bigwedge_{\wp(E)} = \bigcap$ (intersection), $\bigvee_{\wp(E)} = \bigcup$ (union), $\neg_{\wp(E)} A = E \setminus A$ (set complement) and inclusion $\subseteq$ for the order.*

Definition 52 (**e**-models)**.** *Set Σ a signature with basic types containing o and ι and closed λ-terms $\top_\Sigma$, $\bot_\Sigma$, $\wedge_\Sigma$, $\vee_\Sigma$, $\neg_\Sigma$, $\to_\Sigma$, $\forall^\iota_\Sigma$, $\exists^\iota_\Sigma$ having the expected types (*(III.1) *and* (III.2)*).*

*The class **e** of Henkin models on Σ is defined to be the Henkin models $\mathcal{M}$ such that*

1. *$(D_o, [\![\top_\Sigma]\!]^\mathcal{M}, [\![\bot_\Sigma]\!]^\mathcal{M}, [\![\wedge_\Sigma]\!]^\mathcal{M}, [\![\vee_\Sigma]\!]^\mathcal{M}, [\![\neg_\Sigma]\!]^\mathcal{M}, [\![\to_\Sigma]\!]^\mathcal{M}, \bigwedge_\mathbb{B}, \bigvee_\mathbb{B})$ is the complete Boolean algebra of Booleans*
2. *for every $X \in D_{\iota \to o}$, $[\![\forall^\iota_\Sigma]\!]^\mathcal{M}(X) = \bigwedge_\mathbb{B}\{X(a) \mid a \in D_\iota\}$*
3. *for every $X \in D_{\iota \to o}$, $[\![\exists^\iota_\Sigma]\!]^\mathcal{M}(X) = \bigvee_\mathbb{B}\{X(a) \mid a \in D_\iota\}$*

Note that this definition of **e**-models based on Boolean algebras is equivalent to the Tarski's presentation with a satisfaction relation $\models$ presented in section III.1.2.

III.2.2 Extensional semantics

We defined in the previous section logical constants of the object signature $\Sigma_{\mathbf{e}}$ and their denotation. In this section we define linguistic constants of $\Sigma_{\mathbf{e}}$ and the semantic lexicon $\mathfrak{L}_{\mathbf{e}}$ in Tab. III.3.

We write semantic constants in boldface. We suppose that every proper noun $\langle$NAME$\rangle$ present here corresponds to a defined entity $\langle\mathbf{name}\rangle : \iota$. To gain space, only the initial letter of these names are used, e.g. $\mathbf{m}$ for Marie. Verbs like DORT can be reduce to a predicate $\mathbf{sleep} : \iota \to o$ instead of applying to a whole generalized individual $E_{\mathbf{e}} \mathrel{\hat{=}} \mathfrak{L}_{\mathbf{e}}(\mathsf{np}) = (\iota \to o) \to o$. The type of propositions is $T_{\mathbf{e}} \mathrel{\hat{=}} o$.

$$\begin{aligned}
\mathfrak{L}_{\mathbf{e}}(\text{DECL}) = \mathfrak{L}_{\mathbf{e}}(\text{INT}) = \mathfrak{L}_{\mathbf{e}}(\text{QUE}) &\mathrel{\hat{=}} \lambda p^o.\, p \\
\mathfrak{L}_{\mathbf{e}}(\text{ECQ}_1) = \mathfrak{L}_{\mathbf{e}}(\text{SI}) = \mathfrak{L}_{\mathbf{e}}(\text{DECL-INT}) &\mathrel{\hat{=}} \lambda p^o.\, \mathbf{quest}\, p
\end{aligned}$$

$$\begin{aligned}
\mathfrak{L}_{\mathbf{e}}(\text{PEUT-ÊTRE}) &\mathrel{\hat{=}} \lambda p^o.\, \mathbf{M}\, p & \mathfrak{L}_{\mathbf{e}}(\text{DORT}) &\mathrel{\hat{=}} \lambda X^{E_{\mathbf{e}}}.\, X\, (\lambda x^{\iota}.\, \mathbf{sleep}\, x) \\
\mathfrak{L}_{\mathbf{e}}(\text{ECQ}_{2.1}) = \mathfrak{L}_{\mathbf{e}}(\text{ECQ}_{2.2}) &\mathrel{\hat{=}} \lambda F^{E_{\mathbf{e}}o}, H^{(E_{\mathbf{e}}o)o}.\, H\, F & \mathfrak{L}_{\mathbf{e}}(\text{SAIT}_1) &\mathrel{\hat{=}} \lambda p^o, X^{E_{\mathbf{e}}}.\, X\, (\lambda x^{\iota}.\, \mathbf{K}\, x\, p) \\
\mathfrak{L}_{\mathbf{e}}(\text{ECQ}_{3.1}) = \mathfrak{L}_{\mathbf{e}}(\text{ECQ}_{3.2}) &\mathrel{\hat{=}} \lambda H^{A_{\mathbf{e}}o}, F^{A_{\mathbf{e}}}.\, H\, F & \mathfrak{L}_{\mathbf{e}}(\text{SAIT}_2) &\mathrel{\hat{=}} \lambda p^o, X^{E_{\mathbf{e}}}.\, X\, (\lambda x^{\iota}.\, \mathbf{K'}\, x\, p) \\
\mathfrak{L}_{\mathbf{e}}(\langle\text{NAME}\rangle) &\mathrel{\hat{=}} \lambda P^{\iota o}.\, P\, \langle\mathbf{name}\rangle
\end{aligned}$$

$$\begin{aligned}
\mathfrak{L}_{\mathbf{e}}(\text{QUI}) &\mathrel{\hat{=}} \lambda F^{E_{\mathbf{e}}o}.\, \mathbf{exquest}\, (\lambda x^{\iota}.\, F\, (\lambda P^{\iota o}.\, P\, x)) \\
\mathfrak{L}_{\mathbf{e}}(\text{OÙ}) &\mathrel{\hat{=}} \lambda F^{E_{\mathbf{e}}o}, X^{E_{\mathbf{e}}}.\, \mathbf{exquest}\, (\lambda z^{\iota}.\, \mathbf{loc}\, F\, z\, X) \\
\mathfrak{L}_{\mathbf{e}}(\text{CHEZ}) &\mathrel{\hat{=}} \lambda Y^{E_{\mathbf{e}}}, F^{E_{\mathbf{e}}o}, X^{E_{\mathbf{e}}}.\, \exists z^{\iota}.\, (\mathbf{house}\, z) \wedge_{\mathbf{e}} (Y\, (\lambda y^{\iota}.\, \mathbf{own}\, z\, y)) \wedge_{\mathbf{e}} (\mathbf{loc}\, F\, z\, X) \\
\text{where } A_{\mathbf{e}} &\mathrel{\hat{=}} (E_{\mathbf{e}} \to o) \to E_{\mathbf{e}} \to o
\end{aligned}$$

Table III.3 – Lexicon $\mathfrak{L}_{\mathbf{e}}$ from $\Sigma_{\mathbf{D}}$ to $\Sigma_{\mathbf{e}}$

It is not possible to give an interpretation of intentional words, like peut-être or sait. Indeed, peut-être does not change the truth about whether Marie dort or not, but it gives a degree of knowledge about Marie's sleeping. There is no way to model this properly in extensional semantics, so we just use unspecified constants $\mathbf{M} : o \to o$ and $\mathbf{K} : \iota \to o \to o$ for the moment.

Similarly, we are not able here to interpret interrogative meanings. We leave the constants $\mathbf{quest} : o \to o$ (yes/no question), $\mathbf{exquest} : (\iota \to o) \to o$ (existential question) and $\mathbf{K'} : \iota \to o \to o$ (interrogative knowledge) to be detailed later. Nevertheless, it is already possible to tell the semantic function of est-ce que particles. Contrary to ECQ_1 which has actually to add a question raising **quest**, the interpretations of ECQ_2 and ECQ_3 are just structural. They give the interrogative pronoun to the constituent they modify.

Locative is rendered via the constant $\mathbf{loc} : (E_{\mathbf{e}} \to o) \to \iota \to E_{\mathbf{e}}$, taking a generalized predicate F, a place z and a generalized individual X, and expressing that the action F takes place in z and is performed by X.

In our Montagovian approach, this predicate modifier has some flaw yet. There is no guarantee that the denotation of **loc** is monotonic in its arguments. This forbids us, for example, to prove that Marie [dort bien] chez Camille,
[Marie et Jean] dorment chez Camille and Marie dort chez [Camille et Dominique][1] all respectively entail Marie dort chez Camille, which is however true.

To solve this issue, here is how we can enforce monotonicity of modifiers. Either we add axioms, like Montague meaning postulates [48], or we ask that in class **e**, **loc** is upward monotonic in all its argument (see 40 in section I.3.5), and same for the later classes **i** and **q**.

[1] In Marie dort chez [Camille et Dominique], it could be understood that the house is owned by the couple Camille et Dominique and cannot reduce to a proper ownership of each one individually. But this kind of collective readings are left aside in this thesis.

Despite all, this method is not the most elegant because upward monotonicity needs to be specified for certain lexical units manually, independently from the λ-term. A better approach to model locative modifiers is to resort to event semantics (aka. (neo-)Davidsonian semantics). Some proposals to adapt event semantics to ACGs are [60], [29] and [8]. Using a neo-Davidsonian approach is planed as future work.

We provide the $\beta\eta$-reduced object semantics interpretations of the fragment sentences here. Set 1:

(1) a. $\mathbf{sleep}\,\mathbf{m}$
b. $\exists z^{\iota}.\,(\mathbf{house}\,z) \wedge_{\mathbf{e}} (\mathbf{own}\,z\,\mathbf{c}) \wedge_{\mathbf{e}} (\mathbf{loc}\,(\lambda X^{E_{\mathbf{e}}}.\,X\,\mathbf{sleep})\,z\,(\lambda Q^{\iota o}.\,Q\,\mathbf{m}))$

Set 2:

(2) a. $\mathbf{M}\,(\mathbf{sleep}\,\mathbf{m})$
b. $\mathbf{K}\,\mathbf{j}\,(\mathbf{sleep}\,\mathbf{m})$

Set 3 of declarative questions and set 4 of est-ce que questions give the same semantic representations:

(3) a. $\mathbf{quest}\,(\mathbf{sleep}\,\mathbf{m})$
b. $\mathbf{exquest}\,(\lambda x^{\iota}.\,\mathbf{sleep}\,x)$
c. $\mathbf{exquest}\,(\lambda z^{\iota}.\,\mathbf{loc}\,(\lambda X^{E_{\mathbf{e}}}.\,X\,\mathbf{sleep})\,z\,(\lambda Q^{\iota o}.\,Q\,\mathbf{m}))$

Set 5 of embedded questions:

(4) a. $\mathbf{K'}\,\mathbf{j}\,(\mathbf{quest}\,(\mathbf{sleep}\,\mathbf{m}))$
b. $\mathbf{K'}\,\mathbf{j}\,(\mathbf{exquest}\,(\lambda x^{\iota}.\,\mathbf{sleep}\,x))$
c. $\mathbf{K'}\,\mathbf{j}\,(\mathbf{exquest}\,(\lambda z^{\iota}.\,\mathbf{loc}\,(\lambda X^{E_{\mathbf{e}}}.\,X\,\mathbf{sleep})\,z\,(\lambda Q^{\iota o}.\,Q\,\mathbf{m})))$

Let us see some examples of β-reduction.

Sentence (II-12-b) interpreted from (II-17-b) β-reduced into (III-3-b):

$$\begin{aligned}
&\mathfrak{L}_{\mathbf{e}}(\textsc{int}\,(\textsc{qui}\,\textsc{dort}))\\
&\quad = (\lambda p^{o}.\,p)\,(\mathfrak{L}_{\mathbf{e}}(\textsc{qui})\,\mathfrak{L}_{\mathbf{e}}(\textsc{dort}))\\
&\quad \to_{\beta} (\lambda F^{E_{\mathbf{e}}o}.\,\mathbf{exquest}\,(\lambda x^{\iota}.\,F\,(\lambda P^{\iota o}.\,P\,x)))\,(\lambda X^{E_{\mathbf{e}}}.\,X\,(\lambda y^{\iota}.\,\mathbf{sleep}\,y))\\
&\quad \to_{\beta} \mathbf{exquest}\,(\lambda x^{\iota}.\,(\lambda X^{E_{\mathbf{e}}}.\,X\,(\lambda y^{\iota}.\,\mathbf{sleep}\,y))\,(\lambda P^{\iota o}.\,P\,x))\\
&\quad \to_{\beta} \mathbf{exquest}\,(\lambda x^{\iota}.\,(\lambda P^{\iota o}.\,P\,x)\,(\lambda y^{\iota}.\,\mathbf{sleep}\,y))\\
&\quad \to_{\beta} \mathbf{exquest}\,(\lambda x^{\iota}.\,(\lambda y^{\iota}.\,\mathbf{sleep}\,y)\,x)\\
&\quad \to_{\beta} \mathbf{exquest}\,(\lambda x^{\iota}.\,\mathbf{sleep}\,x)
\end{aligned} \tag{III.6}$$

Sentence (II-13-c) interpreted from (II-18-c) $\beta\eta$-reduced into (III-3-c):

$$\begin{aligned}
&\mathfrak{L}_{\mathbf{e}}(\textsc{int}\,(\textsc{ecq}_{3.1}\,(\lambda^{\circ} f^{A_{\mathbf{D}}}.\,f\,\textsc{dort}\,\textsc{Marie})\,\textsc{où}))\\
&\quad \to_{\beta} (\lambda H^{A_{\mathbf{e}}o}, F^{A_{\mathbf{e}}}.\,H\,F)\,(\lambda f^{A_{\mathbf{e}}}.\,f\,\mathfrak{L}_{\mathbf{e}}(\textsc{dort})\,\mathfrak{L}_{\mathbf{e}}(\textsc{Marie}))\,\mathfrak{L}_{\mathbf{e}}(\textsc{où})\\
&\quad \to^{2}_{\beta} (\lambda f^{A_{\mathbf{e}}}.\,f\,\mathfrak{L}_{\mathbf{e}}(\textsc{dort})\,\mathfrak{L}_{\mathbf{e}}(\textsc{Marie}))\,\mathfrak{L}_{\mathbf{e}}(\textsc{où})\\
&\quad \to_{\beta} (\lambda F^{E_{\mathbf{e}}o}, X^{E_{\mathbf{e}}}.\,\mathbf{exquest}\,(\lambda z^{\iota}.\,\mathbf{loc}\,F\,z\,X))\,\mathfrak{L}_{\mathbf{e}}(\textsc{dort})\,\mathfrak{L}_{\mathbf{e}}(\textsc{Marie})\\
&\quad \to^{2}_{\beta} \mathbf{exquest}\,(\lambda z^{\iota}.\,\mathbf{loc}\,\mathfrak{L}_{\mathbf{e}}(\textsc{dort})\,z\,\mathfrak{L}_{\mathbf{e}}(\textsc{Marie}))\\
&\quad \to_{\eta} \mathbf{exquest}\,(\lambda z^{\iota}.\,\mathbf{loc}\,(\lambda X^{E_{\mathbf{e}}}.\,X\,\mathbf{sleep})\,z\,(\lambda Q^{\iota o}.\,Q\,\mathbf{m}))
\end{aligned} \tag{III.7}$$

because $\mathfrak{L}_{\mathbf{e}}(A_{\mathbf{D}}) = A_{\mathbf{e}}$.

III.3 Intentional semantics

Section III.3.1 introduces to intentional semantics by exposing the need of a more expressive system. In section III.3.2, we present intentional logic. We show that we can simulate intentional logic in an object vocabulary of extensional logic by defining intentional connectives as λ-terms over extensional connectives, and a class $\mathbf{i}$ of Henkin models. Finally, we provide the intentional interpretation $\mathfrak{L}_\mathbf{i}$ on λ-terms in section III.3.3.

III.3.1 Beyond truth-conditions

Extensional semantics has the huge drawback to fail to correctly model intentions.

Some adjectives (e.g. $\mathfrak{L}_\mathbf{e}(\textsc{blond}) : (\iota \to o) \to (\iota \to o)$) are intersective, i.e. they modify the meaning of the noun they are qualifying by conjoining their separate lexical meaning. For example, a blond man is a man and a blond person. This may be derived by taking $\mathfrak{L}_\mathbf{e}(\textsc{blond}) = \lambda P^{\iota o}, x^\iota.\, P\, x \wedge_\mathbf{e} \mathbf{blond}\, x$.

However, not all adjectives behave so nicely. If Mary is a gifted lawyer, it does not entail that Mary is a gifted person "in general". For example, if Mary is a gifted lawyer and a baker, having "gifted" intersective would entail that Mary is a gifted baker too, which may not be true.

Suppose now that we are in a world where all lawyers are bakers and vice versa. The predicates **lawyer** and **baker** are co-extensive: they have the same truth-conditions. This would mean that **gifted** (**lawyer**) is the same as **gifted** (**baker**). This is intuitively wrong, because being gifted involves skills that are left implicit and do not only depend on the set of people having the considered position. Such implicit properties are called *intentions*. The meaning of "gifted" depends on the intention of the noun it modifies.

Similarly, sentence modifying adverbs cannot just act on a truth value. Indeed, if it were the case, there would only be $|\mathbb{B}^\mathbb{B}| = 4$ possible distinct sentence modifier meaning. This is clearly not enough to model all natural language modalities like "maybe", "necessarily", "probably", "surely", "unexpectedly",...

A last argument in favor of intentional semantics if the following. In our world where earth is round, the proposition "Earth is round" is co-extensive to "$2 + 2 = 4$". However, it could be that "John knows that $2 + 2 = 4$" is true but that "John knows that Earth is round" is false. Therefore, modal verbs like "know" cannot be properly modeled in extensional semantics.

One way of modeling intentional meanings is to consider several possible worlds [35]. In a world w and for an individual a, **lawyer** $w\, x$ is a truth value. And we want to consider sufficient enough worlds so that there is at least one w and an x where **lawyer** $w\, x$ is not equal to **baker** $w\, x$.

We define the intentional semantic signature $\Sigma_\mathbf{i}$ out of the object semantic one. To avoid confusion, we rename basic types: t for truth values and e for individuals (entities). In typed semantics, possible worlds are introduced as elements of a special set W, and are characterized by the atomic type s. Thus, we have **lawyer** $: s \to e \to t$.

It turns out it is more convenient to put the s-typed argument just before the target type t for systematicity, e.g. $\lambda x^e, w^s.\, \mathbf{lawyer}\, w\, x : e \to s \to t$. This way, we can view $T_\mathbf{i} \hat{=} s \to t$ as the new proposition type.

Formally, $\mathcal{B}_\mathbf{i} = \{s, t, e\}$ and the type morphism of the lexicon $\mathfrak{L}_\mathbf{i}$ from $\Sigma_\mathbf{e}$ to $\Sigma_\mathbf{i}$ is:

$$\mathfrak{L}_\mathbf{i}(\iota) \hat{=} e \qquad \mathfrak{L}_\mathbf{i}(o) \hat{=} s \to t \tag{III.8}$$

III.3.2 Logical structure on worlds

We first introduce intentional logic. Then we define logical constants and their denotation, as we did for $\Sigma_{\mathbf{e}}$. Theorem 10 proves that the class **i** indeed simulates intentional logic.

Definition 53. *A formula of intentional logic is a term φ defined by:*

$$\varphi, \psi ::= \mathbf{P}_n(x_1, ..., x_n) \mid \top_i \mid \bot_i \mid \varphi \wedge_i \psi \mid \varphi \vee_i \psi \mid \neg_i \varphi \mid \varphi \rightarrow_i \psi \mid \forall x.\, \varphi \mid \exists x.\, \varphi \mid \mathbf{M}_i\, \varphi \mid \mathbf{K}_i\, x\, \varphi \quad \text{(III.9)}$$

where x_j an entity variable and $\mathbf{P}_n$ a n-place predicate.

Contrary to what we did in FOL, we do not consider function symbols and first-order terms other than variables because they are not needed.

Modality phenomena are formalized by modal logic. In particular, we are interested here by personal and general knowledge, so we turn to epistemic logic [38]. Traditional denotational semantics of epistemic modal logic uses Kripke frames [42]. This framework is specially suited for typed logic.

Definition 54 (Epistemic Kripke frame)**.** *An epistemic Kripke frame (W, R) is a set W and a relation R on W verifying*

T R *is reflexive*

4 R *is transitive*

5 R *is euclidian: if $w\ \mathsf{R}\ u$ and $w\ \mathsf{R}\ v$, then $u\ \mathsf{R}\ v$*

Note that the usual axiom written **K** in the literature is always valid in Kripke frames. Every relation verifying axioms **T**, **4** and **5** is an equivalence relation, and vice versa.

The accessibility relation R represents what cannot be distinguished. In other words, if $u\ \mathsf{R}\ v$, then we don't know whether we are in world u or in world v.

Definition 55 (Intentional logic)**.** *A model of intentional logic is $\mathcal{M} = (W, D, I, \mathsf{R}, \mathsf{R}_\circ)$, where I maps any $\mathbf{P}_n$ to $I(\mathbf{P}_n) : D^n \rightarrow \wp(W)$ and (W, R) and $(W, \mathsf{R}_\circ(a))$ for every $a \in D$ are epistemic Kripke frames.*

Intentional logic *is the relation $\models_i$ defined by induction on an intentional logic formula φ:*

$$\begin{array}{lll}
\mathcal{M}, g, w \models_i \mathbf{P}_n(x_1, ..., x_n) & \text{if } w \in I(\mathbf{P}_n)(g(x_1), ..., g(x_n)) \\
\mathcal{M}, g, w \models_i \top_i & \\
\mathcal{M}, g, w \not\models_i \bot_i & \\
\mathcal{M}, g, w \models_i \varphi \wedge_i \psi & \text{if } \mathcal{M}, g, w \models_i \varphi \text{ and } \mathcal{M}, g, w \models_i \psi \\
\mathcal{M}, g, w \models_i \varphi \vee_i \psi & \text{if } \mathcal{M}, g, w \models_i \varphi \text{ or } \mathcal{M}, g, w \models_i \psi \\
\mathcal{M}, g, w \models_i \neg_i \varphi & \text{if } \mathcal{M}, g, w \not\models_i \varphi \\
\mathcal{M}, g, w \models_i \varphi \rightarrow_i \psi & \text{if } \mathcal{M}, g, w \models_i \varphi \text{ implies } \mathcal{M}, g, w \models_i \psi \\
\mathcal{M}, g, w \models_i \forall x.\, \varphi & \text{if for all } a \in D, \text{ we have } \mathcal{M}, g[x \mapsto a], w \models_i \varphi \\
\mathcal{M}, g, w \models_i \exists x.\, \varphi & \text{if there exists } a \in D \text{ such that } \mathcal{M}, g[x \mapsto a], w \models_i \varphi \\
\mathcal{M}, g, w \models_i \mathbf{M}_i\, \varphi & \text{if there exists } v \in W \text{ such that } w\ \mathsf{R}\ v \text{ and } \mathcal{M}, g, v \models_i \varphi \\
\mathcal{M}, g, w \models_i \mathbf{K}_i\, x\, \varphi & \text{if for all } v \in W \text{ such that } w\ \mathsf{R}_\circ(g(x))\ v \text{ we have } \mathcal{M}, g, v \models_i \varphi
\end{array} \quad \text{(III.10)}$$

Simulating intentional logic in Henkin models

Similarly to $\Sigma_{\mathbf{e}}$, we want a basis of FOL in $\Sigma_{\mathbf{i}}$ to define richer structures. We equip $\Sigma_{\mathbf{i}}$ with the following constants.

$$\begin{array}{llll} \wedge_{\mathbf{e}} : t \to t \to t & \to_{\mathbf{e}} : t \to t \to t & \top_{\mathbf{e}} : t & \forall^e_{\mathbf{e}} : (e \to t) \to t \quad \forall^s_{\mathbf{e}} : (s \to t) \to t \\ \vee_{\mathbf{e}} : t \to t \to t & \neg_{\mathbf{e}} : t \to t & \bot_{\mathbf{e}} : t & \exists^e_{\mathbf{e}} : (e \to t) \to t \quad \exists^s_{\mathbf{e}} : (s \to t) \to t \end{array} \tag{III.11}$$

We also add to the following constants to $\Sigma_{\mathbf{i}}$:

$$\mathsf{R} : s \to s \to t \qquad \mathsf{R}_\circ : e \to s \to s \to t \tag{III.12}$$

R is the accessibility relation corresponding to common knowledge based on physical likelihood and social norms. $\mathsf{R}_\circ\ x$ is the accessibility relation associated with the personal knowledge of individual x.

Definition 56 (Intentional model)**.** *Set Σ a signature with basic types containing t, s and e and closed λ-terms $\top_\Sigma$, $\bot_\Sigma$, $\wedge_\Sigma$, $\vee_\Sigma$, $\neg_\Sigma$, $\to_\Sigma$, $\forall^e_\Sigma$, $\exists^e_\Sigma$, $\forall^s_\Sigma$, $\exists^s_\Sigma$, R and $\mathsf{R}_\circ$ having the expected types (*(III.11) *and* (III.12)*).*

The class **i** *of Henkin models on Σ is defined to be the Henkin models $\mathcal{M}$ such that*

1. *$(D_t, [\![\top_\Sigma]\!]^{\mathcal{M}}, [\![\bot_\Sigma]\!]^{\mathcal{M}}, [\![\wedge_\Sigma]\!]^{\mathcal{M}}, [\![\vee_\Sigma]\!]^{\mathcal{M}}, [\![\neg_\Sigma]\!]^{\mathcal{M}}, [\![\to_\Sigma]\!]^{\mathcal{M}}, \bigwedge_{\mathbb{B}}, \bigvee_{\mathbb{B}})$ is the complete Boolean algebra of Booleans*
2. *for every $A \in \{e, s\}$, $X \in D_{A\to t}$, $[\![\forall^A_\Sigma]\!]^{\mathcal{M}}(X) = \bigwedge_{\mathbb{B}}\{X(a) \mid a \in D_A\}$*
3. *for every $A \in \{e, s\}$, $X \in D_{A\to t}$, $[\![\exists^A_\Sigma]\!]^{\mathcal{M}}(X) = \bigvee_{\mathbb{B}}\{X(a) \mid a \in D_A\}$*
4. *$(D_s, [\![\mathsf{R}]\!]^{\mathcal{M}})$ and $(D_s, [\![\mathsf{R}_\circ]\!]^{\mathcal{M}}(a))$ for every $a \in D_e$ are epistemic Kripke frames*

We now define the intentional connectives with the help of the extensional ones.

$$\begin{array}{llll} \wedge_{\mathbf{i}} \hat{=} & \lambda p^{T_{\mathbf{i}}}, q^{T_{\mathbf{i}}}, w^s.\, (p\, w) \wedge_{\mathbf{e}} (q\, w) & \top_{\mathbf{i}} \hat{=} & \lambda w^s.\, \top_{\mathbf{e}} \qquad \forall^e_{\mathbf{i}} \hat{=} \lambda P^{eT_{\mathbf{i}}}, w^s.\, \forall x^e.\, P\, x\, w \\ \vee_{\mathbf{i}} \hat{=} & \lambda p^{T_{\mathbf{i}}}, q^{T_{\mathbf{i}}}, w^s.\, (p\, w) \vee_{\mathbf{e}} (q\, w) & \bot_{\mathbf{i}} \hat{=} & \lambda w^s.\, \bot_{\mathbf{e}} \qquad \exists^e_{\mathbf{i}} \hat{=} \lambda P^{eT_{\mathbf{i}}}, w^s.\, \exists x^e.\, P\, x\, w \\ \neg_{\mathbf{i}} \hat{=} & \lambda p^{T_{\mathbf{i}}}, w^s.\, \neg_{\mathbf{e}} (p\, w) & \to_{\mathbf{i}} \hat{=} & \lambda p^{T_{\mathbf{i}}}, q^{T_{\mathbf{i}}}, w^s.\, (p\, w) \to_{\mathbf{e}} (q\, w) \end{array} \tag{III.13}$$

The modal operators can be defined as

$$\begin{array}{lll} \mathbf{M}_{\mathbf{i}} \hat{=} & \lambda p^{T_{\mathbf{i}}}.\, \lambda w^s.\, \exists v^s.\, (\mathsf{R}\ w\, v) \wedge_{\mathbf{e}} (p\, v) & : T_{\mathbf{i}} \to T_{\mathbf{i}} \\ \mathbf{K}_{\mathbf{i}} \hat{=} & \lambda x^e, p^{T_{\mathbf{i}}}.\, \lambda w^s.\, \forall v^s.\, (\mathsf{R}_\circ\ x\, w\, v) \to_{\mathbf{e}} (p\, v) & : e \to T_{\mathbf{i}} \to T_{\mathbf{i}} \end{array} \tag{III.14}$$

To exemplify this, let w be the actual world. "Mary maybe sleeps" is represented by $\mathbf{M}_{\mathbf{i}}\, (\mathbf{sleep}\, \mathbf{m})\, w$, which means: there exists an accessible world v where Mary sleeps in v. Similarly, "John knows that Mary sleeps" is $\mathbf{K}_{\mathbf{i}}\, \mathbf{j}\, (\mathbf{sleep}\, \mathbf{m})\, w$, which means: for every indistinguishable world v, $\mathbf{sleep}\, \mathbf{m}$ holds in v.

Set $\Gamma \vdash_{\mathbf{i}} M : st$ and a model $\mathcal{M}$. Saying that M is intentionally true is saying that for every world $w \in W \hat{=} D_s$, the value of $[\![\Gamma \vdash M]\!]^{\mathcal{M},g}(w)$ is true for every g.

Theorem 10. *Set an* **i***-model $\mathcal{M}$ of $\Sigma_{\mathbf{i}}$. We consider a λ-term φ such that $\Gamma \vdash_{\mathbf{i}} \varphi : st$ and φ is only made up using the λ-terms defined in* (III.13) *and* (III.14)*, variables of type T_E or e and predicates.*

For all g and $w \in D_s$, $[\![\Gamma \vdash \varphi : st]\!]^{\mathcal{M},g}(w) = 1$ iff $\mathcal{M}, g, w \models_i \varphi$

Proof. The proof is straightforward by induction on φ. It is sufficient to remark that the λ-terms of (III.13) and (III.14) implement the left conditions of $\models_i$ in definition 55. □

III.3.3 Intentional lexical meanings

The point of designing a lexicon from $\Sigma_{\mathbf{e}}$ to $\Sigma_{\mathbf{i}}$ is the following. Intentional semantics is better than extensional semantics. However, we might already have spent a lot of time building manually an object semantic vocabulary. Therefore, we would like to embed our vocabulary into intentional semantics, where we can there add other constants to account for purely intentional phenomena. We would also like to preserve the properties of the original vocabulary. This transformation is called a **conservative extension**.

De Groote and Kanazawa showed [25] that there exists a systematic procedure translating an extensional λ-term into an intentional λ-term. This procedure is expressed as a family of λ-terms $\mathbb{E}_A$ for $A \in \mathcal{T}(\mathcal{B}_{\mathbf{e}})$, following the type interpretation in (III.8).[2]

This transformation is proven to be a conservative extension: it preserves truth and entailment.

Theorem 11. *Set* $\vdash_{\mathbf{e}} \varphi : o$ *and* $\vdash_{\mathbf{e}} \psi : o$. *If* $\varphi \models_{\mathbf{e}} \psi$ *then* $\mathbb{E}_o\, \varphi \models_{\mathbf{i}}^{T_{\mathbf{i}}} \mathbb{E}_o\, \psi$

Remark 12. *In their paper, they perform this transformation at the level of denotations. The reformulation we present here is closer to [23]. Moreover, we choose to map individuals to individuals, and not to individual concepts* $s \to e$. *So our embedding* $\mathbb{E}$ *is adapted to have* $\mathbb{E}_\iota\, M = M$ *if* $\Gamma \vdash_{\mathbf{e}} M : \iota$.

It turns out that applying $\mathbb{E}$ to logical constants gives λ-terms that are β-equivalent to the ones in (III.13) and (III.14). We use $\mathbb{E}$ to define the intentional interpretation of the other semantic constants. For example, $\mathfrak{L}_{\mathbf{i}}(\mathbf{sleep})$ uses a new semantic constant $\mathbf{sleep}_{\mathbf{i}} : s \to e \to t$ which denotation now depends on possible worlds. Similarly, $\mathbf{loc}_{\mathbf{i}} : s \to ((et)t)t) \to e \to (et)t \to t$.

The full lexicon is given in Tab. III.4. We use $E_{\mathbf{i}} \mathrel{\hat{=}} (e \to T_{\mathbf{i}}) \to T_{\mathbf{i}}$.

$$
\begin{array}{rcl@{\qquad}rcl}
\mathfrak{L}_{\mathbf{i}}(\top_{\mathbf{e}}) & \hat{=} & \top_{\mathbf{i}} & \mathfrak{L}_{\mathbf{i}}(\mathbf{sleep}) & \hat{=} & \lambda x^e, w^s.\, \mathbf{sleep}_{\mathbf{i}}\, w\, x \\
\mathfrak{L}_{\mathbf{i}}(\bot_{\mathbf{e}}) & \hat{=} & \bot_{\mathbf{i}} & \mathfrak{L}_{\mathbf{i}}(\mathbf{house}) & \hat{=} & \lambda x^e, w^s.\, \mathbf{house}_{\mathbf{i}}\, w\, x \\
\mathfrak{L}_{\mathbf{i}}(\wedge_{\mathbf{e}}) & \hat{=} & \wedge_{\mathbf{i}} & \mathfrak{L}_{\mathbf{i}}(\mathbf{own}) & \hat{=} & \lambda z^e, y^e, w^s.\, \mathbf{own}_{\mathbf{i}}\, w\, z\, y \\
\mathfrak{L}_{\mathbf{i}}(\vee_{\mathbf{e}}) & \hat{=} & \vee_{\mathbf{i}} & \mathfrak{L}_{\mathbf{i}}(\mathbf{M}) & \hat{=} & \lambda p^{T_{\mathbf{i}}}, w^s.\, \mathbf{M}_{\mathbf{i}}\, p\, w \\
\mathfrak{L}_{\mathbf{i}}(\neg_{\mathbf{e}}) & \hat{=} & \neg_{\mathbf{i}} & \mathfrak{L}_{\mathbf{i}}(\mathbf{K}) & \hat{=} & \lambda x^e, p^{T_{\mathbf{i}}}, w^s.\, \mathbf{K}_{\mathbf{i}}\, x\, p\, w \\
\mathfrak{L}_{\mathbf{i}}(\to_{\mathbf{e}}) & \hat{=} & \to_{\mathbf{i}} & \mathfrak{L}_{\mathbf{i}}(\langle \mathbf{name} \rangle) & \hat{=} & \langle \mathbf{name} \rangle \\
\mathfrak{L}_{\mathbf{i}}(\forall^\iota_{\mathbf{e}}) & \hat{=} & \forall^e_{\mathbf{i}} & \mathfrak{L}_{\mathbf{i}}(\exists^\iota_{\mathbf{e}}) & \hat{=} & \exists^e_{\mathbf{i}} \\
\multicolumn{6}{c}{\mathfrak{L}_{\mathbf{i}}(\mathbf{loc}) \mathrel{\hat{=}} \lambda F^{E_{\mathbf{i}} T_{\mathbf{i}}}, z^e, X^{E_{\mathbf{i}}}, w^s.\, \mathbf{loc}_{\mathbf{i}}\, w\, (\lambda Y^{(et)t}.\, F\, (\lambda P^{est}, u^s.\, Y\, (\lambda y^e.\, P\, y\, u))\, w)\, z\, (\lambda Q^{et}.\, X\, (\lambda x^e, v^s.\, Q\, x)\, w)}
\end{array}
$$

Table III.4 – Intentionalization lexicon $\mathfrak{L}_{\mathbf{i}} : \Sigma_{\mathbf{e}} \to \Sigma_{\mathbf{i}}$

Constants related to questions can still not be fully interpreted.

Here are the intentional interpretations of the French examples.

Set 1:

(5) a. $\lambda w^s.\, \mathbf{sleep}_{\mathbf{i}}\, w\, \mathbf{m}$

b. $\lambda w^s.\, \exists z^\iota.\, (\mathbf{house}_{\mathbf{i}}\, w\, z) \wedge_{\mathbf{e}} (\mathbf{own}_{\mathbf{i}}\, w\, z\, \mathbf{c}) \wedge_{\mathbf{e}} (\mathbf{loc}_i\, w\, (\lambda Y^{(et)t}.\, Y\, (\mathbf{sleep}_{\mathbf{i}}\, w))\, z\, (\lambda Q^{et}.\, Q\, \mathbf{m}))$

Set 2:

(6) a. $\lambda v^s.\, \mathbf{M}_{\mathbf{i}}\, (\lambda w.\, \mathbf{sleep}_{\mathbf{i}}\, w\, \mathbf{m})\, v$

[2] Note that in the setting of [23] however, the intentionalization operator is a composition $\mathsf{T} = \mathbb{E} \circ \mathsf{U}$, with U as in definition 66 which can be found later in section IV.2.1.

b. $\lambda v^s.\, \mathbf{K_i}\, \mathbf{j}\, (\lambda w.\, \mathbf{sleep_i}\, w\, \mathbf{m})\, v$

The other sets cannot be interpreted in $\Sigma_{\mathbf{i}}$.

The complex term $\mathfrak{L}_{\mathbf{i}}(\mathbf{loc})$ is due to the interaction of $\mathbb{E}$ and its opposite $\mathbb{P}$ on higher-order types. Let us see with an example that it gives the expected λ-term when further β-reduced. Here is the interpretation of (III-1-b) into (III-5-b).

$$
\begin{aligned}
&\mathfrak{L}_{\mathbf{i}}(\mathbf{loc}\, (\lambda X^{(\iota o)o}.\, X\, \mathbf{sleep})\, z\, (\lambda P^{\iota o}.\, P\, \mathbf{m})) \\
&\quad = \mathfrak{L}_{\mathbf{i}}(\mathbf{loc})\, (\lambda X^{E_{\mathbf{i}}}.\, X\, \mathfrak{L}_{\mathbf{i}}(\mathbf{sleep}))\, z\, (\lambda P^{est}.\, P\, \mathbf{m}) \\
&\quad \to_\beta^3 \lambda w^s.\, \mathbf{loc_i}\, w\, (\lambda Y^{(et)t}.\, (\lambda P^{est}, u^s.\, Y\, (\lambda y^e.\, P\, y\, u))\mathfrak{L}_{\mathbf{i}}(\mathbf{sleep})\ w)\, z\, (\lambda Q^{et}.\, (\lambda P^{est}.\, P\, \mathbf{m})\, (\lambda x^e, v^s.\, Q\, x)\, w) \\
&\quad \to_\beta \lambda w^s.\, \mathbf{loc_i}\, w\, (\lambda Y^{(et)t}.\, (\lambda P^{est}, u^s.\, Y\, (\lambda y^e.\, P\, y\, u))\mathfrak{L}_{\mathbf{i}}(\mathbf{sleep})\ w)\, z\, (\lambda Q^{et}.\, (\lambda x^e, v^s.\, Q\, x)\, \mathbf{m}\, w) \\
&\quad \to_\beta^2 \lambda w^s.\, \mathbf{loc_i}\, w\, (\lambda Y^{(et)t}.\, (\lambda P^{est}, u^s.\, Y\, (\lambda y^e.\, P\, y\, u))\mathfrak{L}_{\mathbf{i}}(\mathbf{sleep})\ w)\, z\, (\lambda Q^{et}.\, Q\, \mathbf{m}) \\
&\quad \to_\beta^2 \lambda w^s.\, \mathbf{loc_i}\, w\, (\lambda Y^{(et)t}.\, Y\, (\lambda y^e.\, \mathfrak{L}_{\mathbf{i}}(\mathbf{sleep})\, y\, w))\, z\, (\lambda Q^{et}.\, Q\, \mathbf{m}) \\
&\quad \to_\beta^2 \lambda w^s.\, \mathbf{loc_i}\, w\, (\lambda Y^{(et)t}.\, Y\, (\lambda y^e.\, \mathbf{sleep_i}\, w\, y))\, z\, (\lambda Q^{et}.\, Q\, \mathbf{m}) \\
&\quad \to_\eta \lambda w^s.\, \mathbf{loc_i}\, w\, (\lambda Y^{(et)t}.\, Y\, (\mathbf{sleep_i}\, w))\, z\, (\lambda Q^{et}.\, Q\, \mathbf{m})
\end{aligned}
\tag{III.15}
$$

IV.
Extending to inquisitive semantics

The last part of this dissertation is dedicated to the semantics of study: inquisitive semantics. In section IV.1 we present inquisitive semantics and adapt it to our setting. In section IV.2, we define formally conservative extensions and extend it to entailment preservation. Finally, we apply the recipe to inquisitive semantics in section IV.3 and discuss the result.

IV.1 Inquisitive semantics

After a quick historical presentation of inquisitive semantics in section IV.1.1, we expose first-order epistemic inquisitive logic (FOEIL) (or just *inquisitive logic*) in section IV.1.2. In section IV.1.3, we design the class of inquisitive models $\mathbf{q}$ in order to simulate FOEIL.

IV.1.1 A representation of interrogative meanings

The great gap of intentional semantics is questions. In a speech, questions may not be present, except rhetorical questions. Nevertheless, they are rampant in dialogues. Erotetics is the philosophical and linguistic study of questions. What questions can we make ? What are the answers of a questions ? Can we define entailment for questions ?

Wiśniewski [61] proposed in 1998 a model called Inferential erotetic logic, based on predicate logic. In this system, interrogation is a game between two players, and playing it amounts to reasoning.

However, representing questions with a different system from statements is questionable. Indeed, interrogative and declarative clauses can both be embedded in interrogative or declarative sentences, like in (IV-1).

(1) a. Jean sait où Marie dort.
b. Est-ce que Jean sait où Marie dort ?
c. Est-ce que Jean sait que Marie dort ?

In the system Hamblin proposed in 1973 [34], a question is represented by the set of its answers, called **alternatives**. This principle is thus called alternative semantics. Polar questions like Est-ce que Marie dort ? have two alternatives: $\lambda w.\, \mathbf{sleep_i}\, w\, \mathbf{m}$ and $\neg_{\mathbf{i}} (\lambda w.\, \mathbf{sleep_i}\, w\, \mathbf{m})$. Open questions have many alternatives. For example, the alternatives of Qui dort ? are all the propositions $\lambda w.\, \mathbf{sleep_i}\, w\, x$ where x is a human being, and the possibility that nobody is actually sleeping.

This framework has been exploited for other uses, e.g. to model focus [55, 3], indefinites [41] and disjunction [1].

Declarative sentences can be represented as questions with just one alternative. However, we cannot use traditional set-theoretic operations, like application and abstraction, on sets of alternatives. For example, sentential conjunction et is usually intersection on meanings, whereas alternative semantics fails to provide a categorematic (simple) operation to model conjunction. Adaptations have been suggested to recover the expected compositionality, e.g. pointwise application. But the lack of a clear-cut entailment relation makes it not elegant enough to be widely used.

In front of the many issues of alternative semantics, Ciardelli, Groenedijk, Roelofsen and Mascarenhas conjointly created **inquisitive semantics** [12] in 2009 and improved it to compositional inquisitive semantics [17] $\mathsf{Inq_B}$.

Instead of just alternatives, the meaning of a question is taken to be all the propositions which are entailed by one of the alternatives. This way, abstraction, application and conjunction keep their traditional denotations. We call **state** S a set of possible worlds, i.e. an intentional proposition. An inquisitive proposition $\mathcal{P}$ is a downward-closed non-empty set of states. The alternatives of $\mathcal{P}$ are the maximal elements of $\mathcal{P}$, where the order $S \leq S'$ is the intentional entailment, i.e. the inclusion of possible worlds $S \subseteq S'$.

A state S *settles* (or supports) an inquisitive formula if there is enough information in S to resolve this formula. If φ encodes a question, S settles φ whenever being in S (at least) answers φ. The inquisitive meaning $\mathcal{P}$ of a formula is the set of states that settle this formula. In the case of declarative sentences, their inquisitive meaning just have one alternative.

This system also handles entailment. A question $\mathcal{P}$ entails a question $\mathcal{Q}$ if knowing the answer of $\mathcal{P}$ allows us to know the answer of $\mathcal{Q}$. For example, suppose we are in a set of worlds where exactly one person sleeps, either Marie, Jean or Camille. Then we have the entailment

$$\text{Qui dort ?} \models \text{Est-ce que Marie dort ?}$$

With $\mathsf{Inq_B}$, we can also establish entailments between interrogative and declarative sentences, and this amounts to set inclusion. For example, an answer to a question always entails this question.

$$\text{Marie dort.} \models \text{Qui dort ?}$$

IV.1.2 First-order epistemic inquisitive logic

Here, we give a formal basis of inquisitive logic [15, 13]. We consider first-order operators ([11] revised with [17]). For simplicity, we do not consider function symbols here, but adding them does not raise any further issue. To account for epistemic behaviors, we take the setting of [16], without their additional E modality.

Definition 57. *A formula of **first-order epistemic inquisitive logic (FOEIL)** is defined by induction:*

$$\varphi, \psi ::= \mathbf{P}_n(x_1, ..., x_n) \mid \top_q \mid \bot_q \mid \varphi \wedge_q \psi \mid \varphi \vee_q \psi \mid \neg_q \varphi \mid \varphi \to_q \psi \mid \forall x.\, \varphi \mid \exists x.\, \varphi \mid \mathbf{M}_q\, \varphi \mid \mathbf{K}_q\, x\, \varphi \tag{IV.1}$$

where x_j an entity variable and $\mathbf{P}_n$ a n-place predicate.

Inquisitive projections are defined as syntactic sugar:

$$!\varphi \mathrel{\hat{=}} \neg\neg\varphi \qquad\qquad ?\varphi \mathrel{\hat{=}} \varphi \vee \neg\varphi \tag{IV.2}$$

Before model-theoretic semantics, let us recall some basic notation to introduce inquisitive epistemic frames.

Definition 58 (Powerset and union set)**.** *The **powerset** $\wp(X)$ of a set X is the set of sets Y such that Y is included in X, viz.*

$$\wp(X) \mathrel{\hat{=}} \{Y \mid Y \subseteq X\}$$

*The **union set** $\bigcup X$ of a set of sets X is the union of the sets belonging to X, viz.*

$$\bigcup X \hat{=} \bigcup_{S \in X} S = \{w \mid \exists S \in X,\ w \in S\}$$

Definition 59. *Fix a set W, which elements are called (possible) **worlds**, and $\mathcal{P} \in \wp(\wp(W))$. We say that $\mathcal{P}$ is downward-closed if for all $S \in \mathcal{P}$ and $R \subseteq S$ we have $R \in \mathcal{P}$.*

*An element of $\wp(W)$ is called a **state**. $\emptyset$ is called the inconsistent state.*

*We write Π_W the set of nonempty downward-closed sets of $\wp(\wp(W))$, called **issues**.*

Definition 60 (Inquisitive epistemic frame)**.** *An inquisitive epistemic frame (W, Ξ) is a set W and a function $\Xi : W \to \Pi_W$ from worlds to issues such that, by defining the **information state** $\sigma(w)$ at world w by*

$$\sigma(w) \hat{=} \bigcup \Xi(w) \tag{IV.3}$$

we have

factivity *for any $w \in W$, $w \in \sigma(w)$*

introspection *for any $w \in W$, if $v \in \sigma(w)$ then $\Xi(v) = \Xi(w)$*

Proposition 11. *If (W, Ξ) is an inquisitive epistemic frame, then (W, R) is an epistemic Kripke frame with*

$$w \mathsf{R} v \text{ if } v \in \sigma(w) \tag{IV.4}$$

Proof. **reflexivity** For $w \in W$, by factivity $w \in \sigma(w)$ so $w \mathsf{R} w$.

symmetry Suppose $w \mathsf{R} v$. As $v \in \sigma(w)$, we have $\Xi(v) = \Xi(w)$ by introspection, so $\sigma(w) = \sigma(v)$ by definition. Using factivity gives us $w \in \sigma(v)$, so $v \mathsf{R} w$.

euclidianness Suppose $w \mathsf{R} v$ and $w \mathsf{R} u$. By the same reasoning, $\Xi(v) = \Xi(w)$, so $\sigma(v) = \sigma(w)$. Therefore, $u \in \sigma(v)$ and $v \mathsf{R} u$.

□

Definition 61 (First-order epistemic inquisitive logic)**.** *A model of FOEIL is $\mathcal{M} = (W, D, I, \Xi, \Xi_\circ)$, where I maps any $\mathbf{P}_n$ to $I(\mathbf{P}_n) : D^n \to \wp(W)$ and (W, Ξ) and $(W, \Xi_\circ(a))$ for every $a \in D$ are inquisitive epistemic frames.*

First-order epistemic inquisitive logic is the relation $\models_q$ defined by induction on a formula φ, for an assignment g and a state $S \subseteq W$:

$$\begin{array}{lll}
\mathcal{M}, g, S \models_q & \mathbf{P}_n(x_1, ..., x_n) & \text{if } S \subseteq I(\mathbf{P}_n)(g(x_1), ..., g(x_n)) \\
\mathcal{M}, g, S \models_q & \top_q & \\
\mathcal{M}, g, S \models_q & \bot_q & \text{if } S = \emptyset \\
\mathcal{M}, g, S \models_q & \varphi \wedge_q \psi & \text{if } \mathcal{M}, g, S \models_q \varphi \text{ and } \mathcal{M}, g, S \models_q \psi \\
\mathcal{M}, g, S \models_q & \varphi \vee_q \psi & \text{if } \mathcal{M}, g, S \models_q \varphi \text{ or } \mathcal{M}, g, S \models_q \varphi \\
\mathcal{M}, g, S \models_q & \neg_q \varphi & \text{if for all } R \subseteq W \text{ such that } \mathcal{M}, g, R \models_q \varphi, \text{ we have } R \cap S = \emptyset \\
\mathcal{M}, g, S \models_q & \varphi \to_q \psi & \text{if for all } R \subseteq S \text{ such that } \mathcal{M}, g, R \models_q \varphi, \text{ we have } \mathcal{M}, g, S \models_q \psi \\
\mathcal{M}, g, S \models_q & \forall x. \varphi & \text{if for all } a \in D, \text{ we have } \mathcal{M}, g[x \mapsto a], S \models_q \varphi \\
\mathcal{M}, g, S \models_q & \exists x. \varphi & \text{if there exists } a \in D \text{ such that } \mathcal{M}, g[x \mapsto a], S \models_q \varphi \\
\mathcal{M}, g, S \models_q & \mathbf{M}_q \varphi & \text{if for all } w \in S \text{ there exists } v \in \sigma(w) \text{ such that } \mathcal{M}, g, \{v\} \models_q \varphi \\
\mathcal{M}, g, S \models_q & \mathbf{K}_q x \varphi & \text{if for all } w \in S, \text{ we have } \mathcal{M}, g, \sigma_{g(x)}(w) \models_q \varphi
\end{array} \tag{IV.5}$$

The definition of the "diamond" epistemic modality $\mathbf{M_q}$ is taken to have $\mathbf{M}_q\,\varphi \equiv_q \neg_q \mathbf{K}_q\,(\neg_q\,\varphi)$.

Proposition 12 (Persistence)**.** *If $\mathcal{M}, g, S \models_q \varphi$, then for all $R \subseteq S$, $\mathcal{M}, g, R \models_q \varphi$*

Proof. We proceed by induction. [15] provides the case of logical connectives. The case of atomic propositions and predicates is obvious. The case of quantifiers follows directly from induction hypothesis. The case of modal operators is also straightforward. □

Here is some vocabulary about inquisitive logic.

Definition 62 (Vocabulary)**.** *Set φ a FOEIL formula. Set a model $\mathcal{M}$ and an assignment g.*
The proposition expressed by φ is $[\varphi]^{\mathcal{M}}_g$, the set of states S such that $\mathcal{M}, g, S \models_q \varphi$.
*The **alternatives** of φ are the maximal elements of $[\varphi]^{\mathcal{M}}_g$ w.r.t. inclusion.*
The truth set of φ is $|\varphi|^{\mathcal{M}}_g \hat{=} \bigcup [\varphi]^{\mathcal{M}}_g$.
φ is called

- *inquisitive if φ has at least two alternatives*[1] *for all $\mathcal{M}, g$*
- ***assertive** (or an assertion) if $[\varphi]^{\mathcal{M}}_g = \{|\varphi|^{\mathcal{M}}_g\}$ for all $\mathcal{M}, g$, i.e. if φ is not inquisitive*
- *informative if $|\varphi|^{\mathcal{M}}_g \neq \wp(W)$ for all $\mathcal{M}, g$*
- *a **question** if φ is not informative*

By proposition 12 and that $\emptyset$ is the minimum of $\wp(W)$ we can conclude that for any formula, $[\varphi]^{\mathcal{M}}_g$ is an issue.

Note that the notion of informativeness and inquisitiveness can adapted to public knowledge to handle properly presuppositions [16]. We leave presupposition treatment for future work.

Proposition 13 (Decomposition between assertion and question)**.** *For every φ we have*

- *φ is an assertion iff $\varphi \equiv_q !\varphi$*
- *φ is a question iff $\varphi \equiv_q ?\varphi$*
- $\varphi \equiv_q !\varphi \wedge_q ?\varphi$

IV.1.3 Inquisitive models

Similarly to intentional logic, we create a signature $\Sigma_{\mathbf{q}}$ where we can express inquisitive semantics, by emulating FOEIL logic with λ-terms.

The basic types of $\Sigma_{\mathbf{q}}$ are $\mathcal{B}_{\mathbf{q}} \hat{=} \{s, t, e\}$ too. We map extensional propositions to inquisitive propositions, of type $T_{\mathbf{q}} = (s \to t) \to t$ (set of states). So the lexicon $\mathfrak{L}_{\mathbf{q}}$ from $\Sigma_{\mathbf{e}}$ to $\Sigma_{\mathbf{q}}$ maps types as follows:

$$\mathfrak{L}_{\mathbf{q}}(\iota) \hat{=} e \qquad\qquad \mathfrak{L}_{\mathbf{q}}(o) \hat{=} (s \to t) \to t \tag{IV.6}$$

We equip $\Sigma_{\mathbf{q}}$ with basic logical connectives, the same as for intentional semantics (III.11) plus quantifiers over states:

[1] Note that in the case W is not finite, it is more accurate to define inquisitiveness as $|\varphi|^{\mathcal{M}}_g \notin [\varphi]^{\mathcal{M}}_g$ for all $\mathcal{M}, g$, because some inquisitive propositions may not have any maximal elements (see [11, sec. 4].

$$\forall^{st}_{\mathbf{e}} : ((s \to t) \to t) \to t \qquad \exists^{st}_{\mathbf{e}} : ((s \to t) \to t) \to t \tag{IV.7}$$

First, we define set-theoretic connectives in (IV.8).

$$\begin{array}{rclcrcl}
\bigcap\limits_{\alpha} & \hat{=} & \lambda A^{\alpha t}, B^{\alpha t}.\, \lambda x^{\alpha}.\, (A\, x) \wedge_{\mathbf{e}} (B\, x) & & \bigcup\limits_{\alpha} & \hat{=} & \lambda A^{\alpha t}, B^{\alpha t}.\, \lambda x^{\alpha}.\, (A\, x) \vee_{\mathbf{e}} (B\, x) \\
\complement\limits_{\alpha} & \hat{=} & \lambda A^{\alpha t}.\, \lambda x^{\alpha}.\, \neg_{\mathbf{e}}(A\, x) & & \emptyset\limits_{\alpha} & \hat{=} & \lambda x^{\alpha}.\, \bot_{\mathbf{e}} \\
\subseteq\limits_{\alpha} & \hat{=} & \lambda A^{\alpha t}, B^{\alpha t}.\, \forall x^{\alpha}.\, (A\, x) \to_{\mathbf{e}} (B\, x) & & =\limits_{\alpha} & \hat{=} & \lambda A^{\alpha t}, B^{\alpha t}.\, (A \subseteq\limits_{\alpha} B) \wedge_{\mathbf{e}} (B \subseteq\limits_{\alpha} A) \\
\{\cdot\}_{\alpha t} & \hat{=} & \lambda x^{\alpha t}.\, \lambda y^{\alpha t}.\, x =\limits_{\alpha} y & & \wp & \hat{=} & \lambda A^{st}.\, \lambda B^{st}.\, B \subseteq\limits_{s} A \\
\bigcup & \hat{=} & \lambda \mathcal{P}^{(st)t}.\, \lambda w^{s}.\, \exists S^{st}.\, (\mathcal{P}\, S) \wedge (S\, w) & & & & \text{for } \alpha \in \{s, st\}
\end{array} \tag{IV.8}$$

The powerset operator $\wp$ acts like a downward-closing operator. We also use $\cdot \neq\limits_{s} \emptyset\limits_{s} \hat{=} \lambda S^{st}.\, \neg_{\mathbf{e}}(S =\limits_{s} \emptyset\limits_{s})$

We define inquisitive connectives (IV.9) as in [7], but we prefer here to give a better intuition of these λ-terms with the help of set-theoretic connectives, as in [17].

$$\begin{array}{rclcrcl}
\wedge_{\mathbf{q}} \hat{=} & \bigcap\limits_{st} & & \top_{\mathbf{q}} \hat{=} & \lambda S^{st}.\, \top_{\mathbf{e}} & & \forall^{e}_{\mathbf{q}} \hat{=} \lambda P^{eT_{\mathbf{q}}}.\, \lambda S^{st}.\, \forall x^{e}.\, P\, x\, S \\
\vee_{\mathbf{q}} \hat{=} & \bigcup\limits_{st} & & \bot_{\mathbf{q}} \hat{=} & \{\emptyset\}_{st} & & \exists^{e}_{\mathbf{q}} \hat{=} \lambda P^{eT_{\mathbf{q}}}.\, \lambda S^{st}.\, \exists x^{e}.\, P\, x\, S \\
& & & \neg_{\mathbf{q}} \hat{=} & \lambda \mathcal{P}^{T_{\mathbf{q}}}.\, \wp \complement\limits_{s} \bigcup \mathcal{P} & \to_{\mathbf{q}} \hat{=} & \lambda \mathcal{P}^{T_{\mathbf{q}}}, \mathcal{Q}^{T_{\mathbf{q}}}.\, \lambda S^{st}.\, ((\wp\, S) \bigcap\limits_{st} \mathcal{P}) \subseteq\limits_{st} \mathcal{Q}
\end{array} \tag{IV.9}$$

And we set $!_{\mathbf{q}} \hat{=} \lambda \mathcal{P}^{T_{\mathbf{q}}}.\, \neg_{\mathbf{q}} \neg_{\mathbf{q}} \mathcal{P}$ and $?_{\mathbf{q}} \hat{=} \lambda \mathcal{P}^{T_{\mathbf{q}}}.\, \mathcal{P} \vee_{\mathbf{q}} \neg_{\mathbf{q}} \mathcal{P}$.
We add inquisitive accessibility relations as constants to $\Sigma_{\mathbf{q}}$:

$$\Xi : s \to (s \to t) \to t \qquad \Xi_{\circ} : e \to s \to (s \to t) \to t \tag{IV.10}$$

The knowledge modal operators can be defined as:

$$\begin{array}{rcll}
\mathbf{M}_{\mathbf{q}} \hat{=} & \lambda \mathcal{P}^{T_{\mathbf{q}}}.\, \wp\, (\lambda w^{s}.\, (\bigcup (\Xi\, w)) \bigcap\limits_{s} (\bigcup \mathcal{P}) \neq\limits_{s} \emptyset\limits_{s}) & : T_{\mathbf{q}} \to T_{\mathbf{q}} \\
\mathbf{K}_{\mathbf{q}} \hat{=} & \lambda x^{e}, \mathcal{P}^{T_{\mathbf{q}}}.\, \wp\, (\lambda w^{s}.\, \mathcal{P}\, (\bigcup (\Xi_{\circ}\, x\, w))) & : e \to T_{\mathbf{q}} \to T_{\mathbf{q}}
\end{array} \tag{IV.11}$$

Now that we have all the ingredients to simulate FOEIL in $\Sigma_{\mathbf{q}}$, we define **q**-models to state and prove our theorem.

Definition 63 (Inquisitive model)**.** *Set Σ a signature with basic types containing t, s and e and closed λ-terms $\top_{\Sigma}$, $\bot_{\Sigma}$, $\wedge_{\Sigma}$, $\vee_{\Sigma}$, $\neg_{\Sigma}$, $\to_{\Sigma}$, $\forall^{e}_{\Sigma}$, $\exists^{e}_{\Sigma}$, $\forall^{s}_{\Sigma}$, $\exists^{s}_{\Sigma}$, $\forall^{st}_{\Sigma}$, $\exists^{st}_{\Sigma}$, Ξ and $\Xi_{\circ}$ having the expected types (*(III.11), (IV.7) *and* (IV.10)*).*

The class **q** *of Henkin models on Σ is defined to be the Henkin models $\mathcal{M}$ such that*

1. *$(D_t, [\![\top_{\Sigma}]\!]^{\mathcal{M}}, [\![\bot_{\Sigma}]\!]^{\mathcal{M}}, [\![\wedge_{\Sigma}]\!]^{\mathcal{M}}, [\![\vee_{\Sigma}]\!]^{\mathcal{M}}, [\![\neg_{\Sigma}]\!]^{\mathcal{M}}, [\![\to_{\Sigma}]\!]^{\mathcal{M}}, \bigwedge_{\mathbb{B}}, \bigvee_{\mathbb{B}})$ is the complete Boolean algebra of Booleans*
2. *for every $A \in \{e, s, st\}$, $X \in D_{A \to t}$, $[\![\forall^{A}_{\Sigma}]\!]^{\mathcal{M}}(X) = \bigwedge_{\mathbb{B}} \{X(a) \mid a \in D_A\}$*
3. *for every $A \in \{e, s, st\}$, $X \in D_{A \to t}$, $[\![\exists^{A}_{\Sigma}]\!]^{\mathcal{M}}(X) = \bigvee_{\mathbb{B}} \{X(a) \mid a \in D_A\}$*
4. *$(D_s, [\![\Xi]\!]^{\mathcal{M}})$ and $(D_s, [\![\Xi_{\circ}]\!]^{\mathcal{M}}(a))$ for every $a \in D_e$ are inquisitive epistemic frames*

Theorem 12. *Set a* $\mathbf{q}$*-model* $\mathcal{M}$ *of* $\Sigma_{\mathbf{q}}$*. We consider a* λ*-term* φ *such that* $\Gamma \vdash_{\mathbf{q}} \varphi : (st)t$ *and* φ *is only made up using the* λ*-terms defined in* (IV.9) *and* (III.14)*, variables of type* $T_{\mathbf{q}}$ *or* e *and predicates.*

For all g *and* $S \in D_{st}$*,* $[\![\Gamma \vdash \varphi : (st)t]\!]^{\mathcal{M},g}(S) = 1$ *iff* $\mathcal{M}, g, S \models_q \varphi$

Proof. The proof is by induction on φ. We just need to prove that the λ-terms M of (IV.9) and (IV.11) implement the left conditions of $\models_q$ in definition 61.

The cases $\wedge_{\mathbf{q}}$, $\vee_{\mathbf{q}}$, $\top_{\mathbf{q}}$, $\bot_{\mathbf{q}}$, $\forall^e_{\mathbf{q}}$ and $\exists^e_{\mathbf{q}}$ are clear from 61.

The cases $\neg_{\mathbf{q}}$ and $\to_{\mathbf{q}}$ can be found in [17] (equation 30 and footnote 21 respectively).

We only provide the details for $\mathbf{M}_{\mathbf{q}}$ and $\mathbf{K}_{\mathbf{q}}$.

Case $\varphi = \mathbf{M}_{\mathbf{q}}\,\psi$ First developing φ gives

$$\varphi \to_\beta \lambda S^{st}.\, \forall w^s.\, (S\, w) \to_{\mathbf{e}} ((\bigcup_s (\Xi\, w)) \underset{s}{\cap} (\bigcup_s \psi) \underset{s}{\neq} \emptyset)$$

By condition 4 of definition 34, we have

$$[\![\Gamma \vdash \varphi : (st)t]\!]^{\mathcal{M},g}(S') = [\![\Gamma, S : st \vdash \forall w^s.\, (S\, w) \to_{\mathbf{e}} ((\bigcup_s (\Xi\, w)) \underset{s}{\cap} (\bigcup_s \psi) \underset{s}{\neq} \emptyset) : t]\!]^{\mathcal{M},g[S\mapsto S']}$$

We reduce this λ-term φ':

$$\begin{aligned}
\varphi' \to_\beta\ & \forall w^s.\, (S\, w) \to_{\mathbf{e}} \neg_{\mathbf{e}}((\bigcup_s (\Xi\, w)) \underset{s}{\cap} (\bigcup_s \psi) \underset{s}{=} \emptyset) \\
\cong_{\mathbf{q}}\ & \forall w^s.\, (S\, w) \to_{\mathbf{e}} \neg_{\mathbf{e}}((\bigcup_s (\Xi\, w)) \underset{s}{\cap} (\bigcup_s \psi) \underset{s}{\subseteq} \emptyset) \\
\to^2_\beta\ & \forall w^s.\, (S\, w) \to_{\mathbf{e}} \neg_{\mathbf{e}}(\forall v^s.\, ((\bigcup_s (\Xi\, w)) \underset{s}{\cap} (\bigcup_s \psi))\, v \to_{\mathbf{e}} \bot_{\mathbf{e}}) \\
\cong_{\mathbf{q}}\ & \forall w^s.\, (S\, w) \to_{\mathbf{e}} (\exists v^s.\, ((\bigcup_s (\Xi\, w)) \underset{s}{\cap} (\bigcup_s \psi))\, v) \\
\to^4_\beta\ & \forall w^s.\, (S\, w) \to_{\mathbf{e}} (\exists v^s.\, (\bigcup_s (\Xi\, w)\, v) \wedge_{\mathbf{e}} (\exists R^{st}.\, (\psi\, R) \wedge_{\mathbf{e}} (R\, v)))
\end{aligned} \tag{IV.12}$$

Suppose $[\![\Gamma \vdash \varphi : (st)t]\!]^{\mathcal{M},g}(S) = 1$. Set $w' \in S'$. There exists v' such that $v' \in \sigma(w') = [\![\Gamma' \vdash \bigcup_s (\Xi\, w) : st]\!]^{\mathcal{M},g'}$ (where $g' \hat{=} g[S \mapsto S', w \mapsto w', v \mapsto v']$, $\Gamma' \hat{=} \Gamma, S : st, w : s$), and for some R' such that $[\![\Gamma', R : st \vdash \psi : (st)t]\!]^{\mathcal{M},g'[R\mapsto R']}(R') = 1$, $v' \in R'$. As none of S, w, v or R appears in ψ because ψ is automatically α-renamed by assumption, we have $[\![\Gamma \vdash \psi : (st)t]\!]^{\mathcal{M},g}(R') = 1$. We can apply the induction hypothesis on ψ. It follows that $\mathcal{M}, g, \{v'\} \models_q \psi$ by persistence (proposition 12).

We have proven that $\mathcal{M}, g, S' \models_q \varphi$.

Now suppose $\mathcal{M}, g, S' \models_q \varphi$. Set $w' \in S'$. There exists $v' \in \sigma(w')$ such that $\mathcal{M}, g, \{v'\} \models_q \psi$. By induction hypothesis we have $[\![\Gamma \vdash \psi : (st)t]\!]^{\mathcal{M},g}(\{v'\}) = 1$. By taking the assignment $R \mapsto \{v'\}$, we clearly have $[\![\Gamma \vdash \varphi : (st)t]\!]^{\mathcal{M},g}(S') = 1$.

Case $\varphi = \mathbf{K}_{\mathbf{q}}\, x\, \psi$ We proceed similarly. There is just to see that the following λ-term is the exact translation of the condition for $\mathcal{M}, g, S' \models_q \varphi$.

$$\varphi\, S \to_\beta \forall w^s.\, (S\, w) \to_{\mathbf{e}} (\psi\, (\bigcup_s (\Xi_\circ\, x\, w))) \tag{IV.13}$$

□

IV.2 Conservative extensions

We would like to design a procedure which creates the lexicon $\mathfrak{L}_{\mathbf{q}}$ on constants, so that the image λ-terms keep the "same" meaning as the intentional ones. Such a procedure is called a conservative extension.

De Groote [23] showed that it is possible to construct a conservative extension if we have the right structure, given a transformation on atomic types. This process works from a basic Montague grammar to intentional semantics [25], dynamic semantics [44] and type raising [23].

In section IV.2.1 we briefly recall this construction and show in section IV.2.2 that, with additional properties on Henkin models, we can extend it to entailment preservation. This result builds on insights and reformulations given by extending the setting to symmetric monoidal closed categories [39].

IV.2.1 De Groote's construction

Recall that a vocabulary $\Sigma = (\mathcal{B}, \mathcal{B}, \mathtt{t})$ is made of basic types $\mathcal{B}$ and constants $\mathcal{C}$ typed by $\mathtt{t} : \mathcal{C} \to \mathcal{T}(\mathcal{B})$ (definition 14). Recall that a lexicon $\mathfrak{L} : \Sigma_1 \to \Sigma_2$ is given by a map of basic types and constants from Σ_1 into types and closed λ-terms of Σ_2 respectively (definition 41). Finally, recall that, given a class $\mathbf{k}$ of Henkin models (definition 34), $\cong_{\mathbf{q}}$ is the logical equivalence, that is, equality of denotation in every $\mathbf{k}$-models.

We can see a vocabulary $\Sigma_{\mathbf{k}}$ as a symmetric monoidal closed category where objects are types and morphisms $A \xrightarrow{M} B$ are λ-terms $x_1 : A_1, ..., x_n : A_n \vdash_{\Sigma_{\mathbf{k}}} M : B$ such that $A = A_1 \otimes ... \otimes A_n$. We can also refine it to a 2-category, where 2-cells are given by $\cong_{\mathbf{k}}$. A lexicon is then a monoidal closed functor.

In the following, we suppose that any vocabulary $\Sigma_{\mathbf{k}}$ involved comes with a class $\mathbf{k}$ of Henkin models. In definition 66, this class is written 2 (the same index as) for signature Σ_2.

Definition 64 (Lexicon transformation)**.** *Set $\Sigma_{\mathbf{0}}, \Sigma_{\mathbf{k}}$ two signatures where we consider a class $\mathbf{k}$ of Henkin models on $\Sigma_{\mathbf{k}}$. Set $\mathfrak{L}_1, \mathfrak{L}_2 : \Sigma_{\mathbf{0}} \to \Sigma_{\mathbf{k}}$ two lexicons.*

A lexicon transformation $\mathbb{E} : \mathfrak{L}_1 \to \mathfrak{L}_2$ is a family of closed λ-terms $\vdash_{\mathbf{k}} \mathbb{E}_A : \mathfrak{L}_1(A) \to \mathfrak{L}_2(A)$ for $A \in \mathcal{T}(\mathcal{B}_2)$ such that for all $\Gamma \vdash_{\mathbf{0}} M : B$,

$$\mathfrak{L}_2(M) \cong_{\mathbf{k}} \mathbb{E}_B \mathfrak{L}_1(M) \tag{IV.14}$$

$$\begin{array}{ccc} & \Sigma_{\mathbf{0}} & \\ \mathfrak{L}_1 & \Big(\mathbb{E} \Big\Downarrow \Big) & \mathfrak{L}_2 \\ & \Sigma_{\mathbf{k}} & \end{array}$$

Lexical transformations are actually natural transformations between lexicons.

Example 7. *Let us use type raising as exposed in [23] to exemplify this definition. Take $\Sigma_{\mathbf{0}} \hat{=} \Sigma_{\mathbf{D}}$ (deep syntax) and $\Sigma_{\mathbf{k}} \hat{=} \Sigma_{\mathbf{e}}$ (extensional semantics). Before the Proper Treatment of Quantification [48], Montague first mapped np to extensional type o, i.e. individuals. Thus we would have*

$$\mathfrak{L}_1(\mathsf{np}) \hat{=} o \qquad \mathfrak{L}_1(\textsc{Marie}) \hat{=} \mathbf{m} \qquad \mathfrak{L}_1(\textsc{dort}) \hat{=} \lambda x^o . \mathbf{sleep}\, x \tag{IV.15}$$

However, it turns out it is better to interpret noun phrases as sets of properties, i.e. type $(o \to \iota) \to \iota$*. That is our lexicon* $\mathfrak{L}_2 \mathrel{\hat{=}} \mathfrak{L}_{\mathbf{e}}$*.*

There exists a lexicon transformation $\mathbb{E}^{tr}$ *from* $\mathfrak{L}_1$ *and* $\mathfrak{L}_{\mathbf{e}}$ *which is called type raising. On first-order and predicate constants, this lexicon transformation is*

$$\mathbb{E}^{tr}_{np}\, M \mathrel{\hat{=}} \lambda P^{o\iota}.\, P\, M \qquad\qquad \mathbb{E}^{tr}_{np \multimap s}\, M \mathrel{\hat{=}} \lambda X^{(o\iota)\iota}.\, X\, (\lambda x^{\iota}.\, M\, x) \tag{IV.16}$$

Definition 65 (Conservative extension)**.** *A lexicon* $\mathfrak{L}_2 : \Sigma_{\mathbf{0}} \to \Sigma_{\mathbf{k}}$ *is a* ***conservative extension*** *of a lexicon* $\mathfrak{L}_1 : \Sigma_{\mathbf{0}} \to \Sigma_{\mathbf{k}}$ *if there exists two lexicon transformations* $\mathbb{E} : \mathfrak{L}_1 \to \mathfrak{L}_2$ *and* $\mathbb{P} : \mathfrak{L}_2 \to \mathfrak{L}_1$ *such that for all* $B \in \mathcal{T}(\mathcal{B}_{\mathbf{0}})$*,* $\mathbb{P}_A\, \mathbb{E}_A \cong_{\mathbf{k}} \lambda x.\, x$*.*

In particular, the denotation of the embedding of a conservative extension is injective.

Requiring that the target signature of $\mathfrak{L}_1$ and $\mathfrak{L}_2$ are the same ensures that there is no constant in the wrong signature. Even if this restriction is mainly due to syntactic reasons, it also serves as a first step to translate entailment between the two object vocabularies of a de Groote's structure.

Definition 66 (De Groote's conservative structure)**.** *Set* $\Sigma_0, \Sigma_{\mathbf{1}}, \Sigma_{\mathbf{2}}$ *three signatures where we consider a class* $\mathbf{1}$ *(resp.* $\mathbf{2}$*) of Henkin models on* $\Sigma_{\mathbf{1}}$ *(resp.* $\Sigma_{\mathbf{2}}$*). Set a lexicon* $\mathfrak{L}_1 : \Sigma_0 \to \Sigma_{\mathbf{1}}$ *and a type morphism* $\mathfrak{L}_2 : \mathcal{B}_0 \to \mathcal{T}(\mathcal{B}_{\mathbf{2}})$ *with* $\mathcal{B}_{\mathbf{1}} \subseteq \mathcal{B}_{\mathbf{2}}$*.*

A de Groote's conservative structure is given by

- *a type transformation* $T : \mathcal{T}(\mathcal{B}_{\mathbf{2}}) \to \mathcal{T}(\mathcal{B}_{\mathbf{2}})$ *coming with 3 operations*

$$\mathsf{U} : A \to TA \qquad \bullet : T(A \to B) \to TA \to TB \qquad \mathsf{C} : (A \to TB) \to T(A \to B) \tag{IV.17}$$

for all $A, B \in \mathcal{T}(\mathcal{B}_{\mathbf{2}})$*, obeying the following laws (*$\bullet$ *is written as an infix operator):*

$$\begin{array}{rcl} (\mathsf{U}\, f) \bullet (\mathsf{U}\, x) & \cong_2 & \mathsf{U}\, (f\, x) \\ \mathsf{C}\, (\lambda x.\, \mathsf{U}\, (f\, x)) & \cong_2 & \mathsf{U}\, f \end{array} \tag{IV.18}$$

- *for every atomic type* $a \in \mathcal{B}_0$*, two* λ*-terms: the* ***embedding*** $\mathbb{E}_a : T\mathfrak{L}_1(a) \to \mathfrak{L}_2(a)$ *and the* ***projection*** $\mathbb{P}_a : \mathfrak{L}_2(a) \to T\mathfrak{L}_1(a)$*, such that*

$$\mathbb{P}_a\, (\mathbb{E}_a\, (\mathsf{U}\, M)) \cong_{\mathbf{2}} \mathsf{U}\, M \tag{IV.19}$$

for all M*.*

For example, type raising is a conservative extension which can be defined by a de Groote's conservative structure. Intentionalization (from the identity on $\Sigma_{\mathbf{e}}$ to $\mathfrak{L}_{\mathbf{i}} : \Sigma_{\mathbf{e}} \to \Sigma_{\mathbf{i}}$) is also a conservative extension which can be described by a de Groote's conservative structure [23].

Theorem 13. *If we have a de Groote's conservative structure, then we can define lexicon transformations for any type by*

$$\begin{array}{rcl} \mathbb{E}_{A \to B} M & \mathrel{\hat{=}} & \lambda x^{\mathfrak{L}_2(A)}.\, \mathbb{E}_B\, (M \bullet (\mathbb{P}_A\, x)) \\ \mathbb{P}_{A \to B} M & \mathrel{\hat{=}} & \mathsf{C}\, (\lambda x^{\mathfrak{L}_1(A)}.\, \mathbb{P}_B\, (M\, (\mathbb{E}_A\, (\mathsf{U}\, x)))) \end{array} \tag{IV.20}$$

By taking $\mathcal{C}_{\mathbf{2}} \mathrel{\hat{=}} \mathcal{C}_{\mathbf{1}}$ *and defining,* $\mathfrak{L}_2(c) \mathrel{\hat{=}} \mathbb{E}_{\mathfrak{t}(c)}\, (\mathsf{U}\, \mathfrak{L}_1(c)) : \mathfrak{L}_2(\mathfrak{t}(c))$ *for every constant* $c \in \mathcal{C}_{\mathbf{0}}$*, we build a signature* $\Sigma_{\mathbf{2}}$ *such that the lexicon* $\mathfrak{L}_2$ *is a conservative extension of* $\mathfrak{I} \circ \mathfrak{L}_1$*, where* $\mathfrak{I}$ *is the injection lexicon from* $\Sigma_{\mathbf{1}}$ *to* $\Sigma_{\mathbf{2}}$*.*

Proposition 14 (Compositionality)**.** *If* $(\mathbb{E}, \mathbb{P})$ *is a de Groote's conservative extension, then for all right* M, N

$$(\mathbb{E}\, (\mathsf{U}\, M))\, (\mathbb{E}\, (\mathsf{U}\, N)) \cong_{\mathbf{2}} \mathbb{E}\, (\mathsf{U}\, (M\, N)) \tag{IV.21}$$

Remark 13. *We can see* T *as a monoidal closed monad.* U *is the unit,* $\bullet$ *is the closed natural transformation and* C *is a closed strength [30]. The existence of a conservative extension was generalized to the existence of a forgetful functor between refinement systems by [39].*

IV.2.2 Entailment-conservative extension

Contrary to [23] where syntactic lexicon transformations are used to prove theorem 13, we bring here a novel and stronger proof, closer to the idea of [25], but using additional conditions.

Definition 67 (Monotonic de Groote's conservative structure)**.** *A monotonic de Groote's conservative structure is a de Groote's conservative structure such that*

1. U, C *and for every* $a \in \mathcal{B}_{\mathbf{0}}$ $\mathbb{E}_a$ *and* $\mathbb{P}_a$ *are upward monotonic in their respective arguments.* $\bullet$ *is upward monotonic in its first argument.*
2. *for all* M, N, *if* $M \models_{\mathbf{1}}^{\mathfrak{L}_1(A)} N$ *then* $\mathfrak{I}(M) \models_{\mathbf{2}}^{\mathfrak{I}(\mathfrak{L}_1(A))} \mathfrak{I}(N)$

The last conditions ensures that the class of $\mathbf{2}$-models is not "too small" to interpret λ-terms of $\Sigma_{\mathbf{1}}$. In this subsection, we assume $\mathfrak{I}$ might not be the identity on every constant, as it is the case for inquisitivation.

The following theorem and the application to inquisitivation are the main contribution of this thesis.

Theorem 14. *The conservative extension of a monotonic de Groote's conservative extension preserves entailment, i.e. for all* $M, N : A$*,*

$$\text{if } M \models_{\mathbf{1}}^{\mathfrak{L}_1(A)} N, \text{ then } \mathbb{E}_A\left(\mathsf{U}\,\mathfrak{I}(M)\right) \models_{\mathbf{2}}^{\mathfrak{L}_2(A)} \mathbb{E}_A\left(\mathsf{U}\,\mathfrak{I}(N)\right)$$

Given a monotonic de Groote's conservative structure, the theorem is a consequence of the following lemma.

Lemma 4. *Set* $\Gamma \vdash_{\mathbf{2}} M : T\mathfrak{L}_1(A)$, $\Delta \vdash_{\mathbf{2}} N : T\mathfrak{L}_1(A)$, $\Gamma' \vdash_{\mathbf{2}} M' : \mathfrak{L}_2(A)$, $\Delta' \vdash_{\mathbf{2}} N' : \mathfrak{L}_2(A)$*. Then*

1. *if* $M \models_{\mathbf{2}}^{T\mathfrak{L}_1(A)} N$, *then* $\mathbb{E}_A\, M \models_{\mathbf{2}}^{\mathfrak{L}_2(A)} \mathbb{E}_A\, N$
2. *if* $M' \models_{\mathbf{2}}^{\mathfrak{L}_2(A)} N'$, *then* $\mathbb{P}_A\, M' \models_{\mathbf{2}}^{T\mathfrak{L}_1(A)} \mathbb{P}_A\, N'$

Proof. By induction on $A \in \mathcal{T}(\mathcal{B}_0)$.

Case $A \in \mathcal{B}_0$ **:**

1. As $\mathbb{E}_A$ is upward monotonic in its argument, we can deduce that $\mathbb{E}_A\, M \models_{\mathbf{2}}^{\mathfrak{L}_2(A)} \mathbb{E}_A\, N$ by proposition 8.
2. The case $\mathbb{P}_A$ is similar.

Case $A = B \to C$ **:**

1. Set $b \in D_{\mathfrak{L}_2(B)}$. By monotonicity of $\bullet$ in its first arguments and the one of $\mathbb{E}_C$ by induction hypothesis,

   $$[\![\mathbb{E}_C\,(N \bullet (\mathbb{P}_B\, x))]\!]^{\mathcal{M},g[x\mapsto b]} \leq_{\mathfrak{L}_2(C)} [\![\mathbb{E}_C\,(M \bullet (\mathbb{P}_B\, x))]\!]^{\mathcal{M},g[x\mapsto b]}$$

   We proved that $[\![\mathbb{E}_A\, M]\!]^{\mathcal{M},g} \leq_{\mathfrak{L}_2(A)} [\![\mathbb{E}_A\, N]\!]^{\mathcal{M},g}$, so $\mathbb{E}_A\, M \models_{\mathbf{2}}^{A} \mathbb{E}_A\, N$.
2. The $\mathbb{P}_A$ case works similarly, by upward monotonicity of the application, $\mathbb{P}_C$ (by induction hypothesis) and C.

□

Proof of the theorem. Suppose $M \models_{\mathbf{1}}^{\mathfrak{L}_1(A)} N$. We have $\mathfrak{I}(M) \models_{\mathbf{2}}^{\mathfrak{L}_1(A)} \mathfrak{I}(N)$ by item 2 of definition 67. Then by upward monotonicity of U and lemma 4, we have the expected result. □

Corollary 2. *If* $M \cong_{\mathbf{1}} N$, *then* $\mathbb{E}\left(\mathsf{U}\,\mathfrak{I}(M)\right) \cong_{\mathbf{2}} \mathbb{E}\left(\mathsf{U}\,\mathfrak{I}(N)\right)$.

IV.3 Inquisitivation

We design a de Groote's conservative structure for inquisitivation in section IV.3.1. In section IV.3.2, we discuss the relationships between the image of intentional logical connectives and their inquisitive correspondents. Finally, we provide the interpretation of linguistic constants and on the French fragment in section IV.3.3.

IV.3.1 Application

We use de Groote's conservative extension to embed the intentional lexicon into the inquisitive one $\mathfrak{L}_{\mathbf{q}} : \Sigma_{\mathbf{e}} \to \Sigma_{\mathbf{q}}$. The global diagram of this thesis is drawn in Fig. IV.1

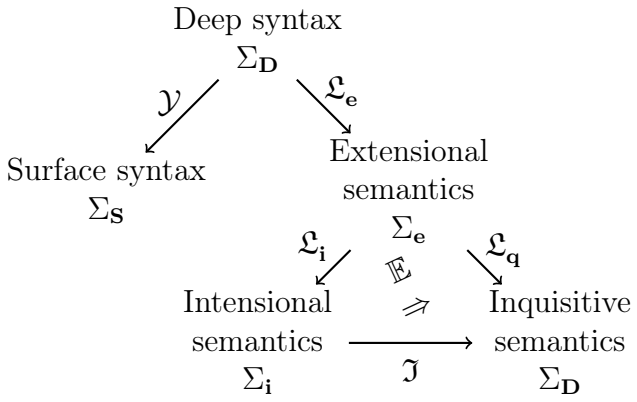

Figure IV.1 – All vocabularies and lexicons considered in this thesis

In inquisitive semantics, a declarative sentence is modeled by an issue $\mathcal{P}$ which has a maximum S, corresponding to the intentional meaning of the sentence. In other words, $\mathcal{P} = \wp(S)$, which is the same as saying $\bigcup \mathcal{P} = S$. That's why we use these operations to define the embedding and projection on atomic types.

Proposition 15. *We have the adjunction $\bigcup \dashv \wp$ (i.e. $\bigcup \mathcal{P} \subseteq S$ iff $\mathcal{P} \subseteq \wp(S)$).*

Proof. Take W a set, $S \subseteq W$ and $\mathcal{P} \in \wp(\wp(W))$.

Suppose $\bigcup \mathcal{P} \subseteq S$. Set $R \in \mathcal{P}$. As $R \subseteq \bigcup \mathcal{P}$, $R \subseteq S$. Therefore, $R \in \wp(S)$.

Now suppose $\mathcal{P} \subseteq \wp(S)$. Set $w \in \bigcup \mathcal{P}$. There exists $R \in \mathcal{P}$ such that $w \in R$. As $R \in \wp(S)$, we conclude that $w \in S$.

We proved that $\bigcup \mathcal{P} \subseteq S$ iff $\mathcal{P} \subseteq \wp(S)$, that is, the (monotonic) Galois connection $\bigcup \dashv \wp$. □

Corollary 3. *$\bigcup$ and $\wp$ are upward monotonic.*

Proposition 16. $\bigcup \wp = \mathsf{id}$

Proof. Take W a set and $S \subseteq W$.

Set $w \in S$. $\{w\} \in \wp(S)$ so $w \in \bigcup(\wp(S))$.

Conversely, set $w \in \bigcup(\wp(S))$. There exists $R \in \wp(S)$ such that $w \in R$, so clearly $w \in S$. □

Thus, an equivalent λ-term for the assertion projection is $!_{\mathbf{q}} \cong_{\mathbf{q}} \lambda \mathcal{P}^{T_{\mathbf{q}}}.\ \wp \bigcup \mathcal{P}$. Moreover, $[\![!_{\mathbf{q}}]\!]^{\mathcal{M},g}$ is the unit of the adjunction.

Definition 68 (Inquisitivation)**.** ***Inquisitivation*** *is the de Groote's conservative extension defined by*

$$\begin{array}{rcl} T\alpha \mathrel{\hat{=}} \alpha \\ \mathsf{U}\, S \mathrel{\hat{=}} S \\ f \bullet x \mathrel{\hat{=}} f\, x \\ \mathsf{C}\, f \mathrel{=\!\hat{=}} f \end{array} \qquad \begin{array}{rcl} \mathbb{E}_\iota\, x \mathrel{\hat{=}} x \\ \mathbb{P}_\iota\, x \mathrel{\hat{=}} x \\ \mathbb{E}_o\, S \mathrel{\hat{=}} \wp\, S \\ \mathbb{P}_o\, \mathcal{P} \mathrel{\hat{=}} \bigcup \mathcal{P} \end{array} \qquad \mathfrak{I}(c) \mathrel{\hat{=}} \begin{cases} c & \text{if } c \in \mathcal{C}_1 \setminus \{\mathsf{R}, \mathsf{R}_\circ\} \\ \bigcup \Xi & \text{if } c = \mathsf{R} \\ \bigcup \Xi_\circ & \text{if } c = \mathsf{R}_\circ \end{cases} \tag{IV.22}$$

The conditions are clearly verified.

Simplifying the λ-terms of inquisitivation yields the **embedding** and **projection** of Fig. IV.2.

$$\begin{array}{rcl} \mathbb{E}_\iota\, x & = & x \\ \mathbb{P}_\iota\, x & = & x \\ \mathbb{E}_o\, S & = & \wp\, S \\ \mathbb{P}_o\, \mathcal{P} & = & \bigcup \mathcal{P} \\ \mathbb{E}_{A \to B}\, M & = & \lambda x^{\mathfrak{L}_{\mathbf{q}}(A)}.\, \mathbb{E}_B\, (M\, (\mathbb{P}_A\, x)) \\ \mathbb{P}_{A \to B}\, M & = & \lambda x^{\mathfrak{L}_{\mathbf{i}}(A)}.\, \mathbb{P}_B\, (M\, (\mathbb{E}_A\, x)) \end{array}$$

Figure IV.2 – Embedding and projection of inquisitivation

Proposition 17. *Inquisitivation is a monotonic de Groote's conservative extension.*

Proof. Upward monotonicity is trivial or consequence of proposition 15. There is just condition 2 to prove.

Suppose $M \models_{\mathbf{i}}^{\mathfrak{L}_1(A)} N$. Set a $\mathbf{q}$-model $\mathcal{M}$ and g. We have by the definitions of $\mathbf{i}$ and $\mathbf{q}$ models and thanks to proposition 11 that $\mathcal{M}$ is also an $\mathbf{i}$ model. Therefore $[\![\mathfrak{I}(M)]\!]^{\mathcal{M},g} \leq_{\mathfrak{L}_1(A)} [\![\mathfrak{I}(N)]\!]^{\mathcal{M},g}$ because $\mathfrak{I}$ transforms epistemic relations as in proposition 11. Consequently, $\mathfrak{I}(M) \models_{\mathbf{q}}^{\mathfrak{L}_1(A)} \mathfrak{I}(N)$. □

IV.3.2 Transformation of logical connectives

The inquisitive lexicon is given by inquisitivation of the intentional lexicon. For every constant $c \in \mathcal{C}_0$, we take

$$\mathfrak{L}_{\mathbf{q}}(c) \mathrel{\hat{=}} \mathbb{E}_{\mathfrak{t}(c)}\, \mathfrak{L}_{\mathbf{i}}(c)$$

For some logical connectives $\star$, the inquisitivation of the intentional interpretation of $\star$ coincides with the inquisitive interpretation of $\star$ as defined in section IV.1.3. For some, they only coincide on the image of the embedding $\mathbb{E}$, i.e. on assertions. Disjunction and existential quantification, however, do not coincide respectively.

Let us formalize these remarks.

Proposition 18.

$$\mathbb{E}\, \top_{\mathbf{i}} \;\cong_{\mathbf{q}}\; \top_{\mathbf{q}} \qquad \mathbb{E}\, \bot_{\mathbf{i}} \;\cong_{\mathbf{q}}\; \bot_{\mathbf{q}} \qquad \mathbb{E}\, \neg_{\mathbf{i}} \;\cong_{\mathbf{q}}\; \neg_{\mathbf{q}} \qquad \mathbb{E}\, \mathbf{M}_{\mathbf{i}} \;\cong_{\mathbf{q}}\; \mathbf{M}_{\mathbf{q}} \tag{IV.23}$$

Proof. Straightforward by writing the λ-term. □

Note that we clearly see on $\neg_{\mathbf{q}}$ how $\mathbb{E}$ acts by transporting the structure of $\neg_{\mathbf{i}}$ from $\Sigma_{\mathbf{i}}$ to $\Sigma_{\mathbf{q}}$.

Proposition 19. *If the denotation of the λ-terms $\mathcal{P}$, $\mathcal{Q}$ and $P\,M$ are assertions in all models for all $\Gamma \vdash_{\mathbf{q}} M : e$, then*

$$\begin{array}{rcl@{\qquad\qquad}rcl} \mathcal{P}\,(\mathbb{E}\wedge_{\mathbf{i}})\,\mathcal{Q} & \cong_{\mathbf{q}} & \mathcal{P}\wedge_{\mathbf{q}}\mathcal{Q} & \mathcal{P}\,(\mathbb{E}\ \to_{\mathbf{i}})\,\mathcal{Q} & \cong_{\mathbf{q}} & \mathcal{P}\to_{\mathbf{q}}\mathcal{Q} \\ (\mathbb{E}\,\forall_{\mathbf{i}})\,P & \cong_{\mathbf{q}} & \forall_{\mathbf{q}}\,P & (\mathbb{E}\,\mathbf{K}_{\mathbf{i}})\,x\,\mathcal{P} & \cong_{\mathbf{q}} & \mathbf{K}_{\mathbf{q}}\,x\,\mathcal{P} \end{array} \tag{IV.24}$$

Proof. First remark that the denotation of $\mathcal{P}$ is an assertion iff $\mathcal{P} \cong_{\mathbf{q}} \wp\,(\bigcup \mathcal{P})$. Suppose the denotation of $\mathcal{P}$ and $\mathcal{Q}$ are assertions.

The case of $\wedge_{\mathbf{i}}$ is due to

$$\wp\,(\bigcup \mathcal{P}) \underset{st}{\cap} \wp\,(\bigcup \mathcal{Q}) \cong_{\mathbf{q}} \wp\,((\bigcup \mathcal{P}) \underset{s}{\cap} (\bigcup \mathcal{Q}))$$

The case of $\forall_{\mathbf{i}}$ is similar as it is an intersection on all individuals. Indeed, we have for all $\mathcal{M}, g$

$$\begin{array}{rcl} [\![\forall_{\mathbf{q}}\,P]\!]^{\mathcal{M},g} = & S' \mapsto \bigwedge_{\mathbb{B}}\{a \in D_e \mid [\![P]\!]^{\mathcal{M},g}(a)(S')\} \\ = & \{S' \mid \text{ for all } a \in D_e, S' \in [\![P]\!]^{\mathcal{M},g}(a)\} \\ = & \bigcup_{a\in D_e} [\![P]\!]^{\mathcal{M},g}(a) \end{array} \tag{IV.25}$$

The case of $\mathbf{K}_{\mathbf{i}}$ is due to

$$S \underset{st}{\subseteq} \bigcup (\wp\,(\bigcup \mathcal{P})) \cong_{\mathbf{q}} (\wp \bigcup \mathcal{P})\,S$$

for any S.

For implication, combining the previous equivalences allows us to derive

$$\begin{array}{rcl} \mathcal{P}\to_{\mathbf{q}}\mathcal{Q} \cong_{\mathbf{q}} & \lambda S^{st}.\ \wp\ S \underset{st}{\cap} \wp\,(\bigcup \mathcal{P}) \underset{st}{\subseteq} \wp\,(\bigcup \mathcal{Q}) \\ \cong_{\mathbf{q}} & \lambda S^{st}.\ \wp\,(S \underset{s}{\cap} (\bigcup \mathcal{P})) \underset{st}{\subseteq} \wp\,(\bigcup \mathcal{Q}) \\ \cong_{\mathbf{q}} & \lambda S^{st}.\ S \underset{s}{\cap} \bigcup \mathcal{P} \underset{s}{\subseteq} \bigcup \mathcal{Q} \\ =_{\beta} & \lambda S^{st}.\ \forall w^{s}.\ (S\,w \wedge_{\mathbf{e}} (\bigcup \mathcal{P})\,w) \to_{\mathbf{e}} (\bigcup \mathcal{Q})\,w \\ \cong_{\mathbf{q}} & \lambda S^{st}.\ \forall w^{s}.\ S\,w \to_{\mathbf{e}} (\neg_{\mathbf{e}}(\bigcup \mathcal{P})\,w \vee_{\mathbf{e}} (\bigcup \mathcal{Q})\,w) \\ =_{\beta} & \lambda S^{st}.\ \wp\,((\complement_{s}(\bigcup \mathcal{P})) \underset{s}{\cup} \bigcup \mathcal{Q}) \\ =_{\beta} & \mathcal{P}\,(\mathbb{E}\ \to_{\mathbf{i}})\,\mathcal{Q} \end{array} \tag{IV.26}$$

□

To say it with words for the case of conjunction, interpreting and as intentional conjunction is the same as interpreting it as inquisitive conjunction for declarative sentences. Therefore, we can redefine the interpretation of these logical connectives to be their inquisitive correspondent, as in (IV.27). It does not change the behavior of the connectives on declarative meanings but reuses the connectives of FOEIL.

$$\mathfrak{L}_{\mathbf{q}}(\wedge_{\mathbf{e}}) \mathrel{\hat{=}} \wedge_{\mathbf{q}} \qquad \mathfrak{L}_{\mathbf{q}}(\to_{\mathbf{e}}) \mathrel{\hat{=}} \to_{\mathbf{q}} \qquad \mathfrak{L}_{\mathbf{q}}(\forall_{\mathbf{e}}) \mathrel{\hat{=}} \forall_{\mathbf{q}} \qquad \mathfrak{L}_{\mathbf{q}}(\mathbf{K}) \mathrel{\hat{=}} \mathbf{K}_{\mathbf{q}} \tag{IV.27}$$

Another special case is for the knowledge operator $\mathbf{K}$. Embedding this intentional modality gives a modality, which acts as the intentional one on non-inquisitive issues. This proves that the inquisitive modality defined $\mathbf{K}_{\mathbf{q}}$ is indeed a "natural" conservative extension of $\mathbf{K}_{\mathbf{i}}$, as suggested in [16].

On the contrary, intentional disjunction and existential quantification cannot be modeled by their sole inquisitive equivalent. We have

$$\begin{aligned} \mathcal{P}\,(\mathbb{E}\vee_{\mathbf{i}})\,\mathcal{Q} &=_\beta \wp\,((\bigcup\mathcal{Q})\underset{s}{\cup}(\bigcup\mathcal{R})) \\ (\mathbb{E}\,\exists_{\mathbf{i}})\,P &=_\beta \wp\,(\lambda w^s.\,\exists x^e.\,\bigcup(\mathcal{P}\,x)\,w) \end{aligned} \tag{IV.28}$$

Set $\mathcal{M}$ a model and suppose D_s has at least two different elements w and w'. Let's call I the issue $\wp(\{w\}) = \{\{w\},\emptyset\}$ and I' the issue $\wp(\{w'\})$. I and I' are assertions, but we have

$$[\![\mathbb{E}\vee_{\mathbf{i}}]\!]^{\mathcal{M},\emptyset}(I)(I') = \wp(\{w,w'\}) \neq \{\{w\},\{w'\},\emptyset\} = I\cup I' = [\![\vee_{\mathbf{q}}]\!]^{\mathcal{M},\emptyset}(I)(I') \tag{IV.29}$$

A similar counterexample for $\exists_{\mathbf{i}}$ works by taking, if D_e has at least two elements and $a\in D_e$,

$$P : b \mapsto \begin{cases} I & \text{if } b = a \\ I' & \text{if } b \neq a \end{cases} \tag{IV.30}$$

Nevertheless, we can use the the assertion projection $!_{\mathbf{q}}$ to express $\mathfrak{L}_{\mathbf{q}}(\vee_{\mathbf{e}})$ with inquisitive connectives only.

Proposition 20.

$$\mathbb{E}\vee_{\mathbf{i}} \cong_{\mathbf{q}} \lambda\mathcal{P}^{T_{\mathbf{q}}},\mathcal{Q}^{T_{\mathbf{q}}}.\,!_{\mathbf{q}}\,(\mathcal{P}\vee_{\mathbf{q}}\mathcal{Q}) \qquad\qquad \mathbb{E}\,\exists_{\mathbf{i}} \cong_{\mathbf{q}} \lambda P^{eT_{\mathbf{q}}}.\,!_{\mathbf{q}}\,(\exists_{\mathbf{q}}\,P) \tag{IV.31}$$

Proof. Left to the reader. □

IV.3.3 Transformation of linguistic constants

Linguistics constants are interpreted with the help of $\mathbb{E}$ in Tab. IV.1. We use $E_{\mathbf{q}} \hat{=} (e\to T_{\mathbf{q}})\to T_{\mathbf{q}}$. We also define constants to interpret special inquisitive meanings.

In particular, both interpretations of the verb **sait** are now the same, because $\mathfrak{L}_{\mathbf{q}}(\mathbf{K}) = \mathfrak{L}_{\mathbf{q}}(\mathbf{K}')$. The difference of result lies now at the level of the embedded clause, which either gives an assertion or a question.

$$\begin{aligned} \mathfrak{L}_{\mathbf{q}}(\mathbf{sleep}) &=_\beta \lambda x^e.\,\wp\,(\lambda w^s.\,\mathbf{sleep}_{\mathbf{i}}\,w\,x) & \mathbf{quest}_{\mathbf{q}} &\hat{=} ?_{\mathbf{q}} \\ \mathfrak{L}_{\mathbf{q}}(\mathbf{house}) &=_\beta \lambda x^e.\,\wp\,(\lambda w^s.\,\mathbf{house}_{\mathbf{i}}\,w\,x) & \mathbf{exquest}_{\mathbf{q}} &\hat{=} \lambda\mathcal{P}^{eT_{\mathbf{q}}}.\,?_{\mathbf{q}}\,(\exists_{\mathbf{q}}\,P) \\ \mathfrak{L}_{\mathbf{q}}(\mathbf{own}) &=_\beta \lambda z^e,y^e.\,\wp\,(\lambda w^s.\,\mathbf{house}_{\mathbf{i}}\,w\,z\,y) & \mathfrak{L}_{\mathbf{q}}(\mathbf{K}') &\hat{=} \mathbf{K}_q \\ \mathfrak{L}_{\mathbf{q}}(\langle\mathbf{name}\rangle) &= \langle\mathbf{name}\rangle & & \end{aligned}$$

$$\begin{aligned} \mathfrak{L}_{\mathbf{q}}(\mathbf{loc}) =_\beta \lambda P^{E_{\mathbf{q}}T_{\mathbf{q}}},z^e,X^{E_{\mathbf{q}}}.\ \wp(\ &\lambda w^s.\ \mathbf{loc}_{\mathbf{i}}\,w \\ &(\lambda Y^{(et)t}.\,\bigcup(P\,(\lambda f^{eT_{\mathbf{q}}}.\,\wp\,(\lambda u^s.\,Y\,(\lambda y^e.\,\bigcup(f\,y)\,u))))\,w) \\ &z \\ &(\lambda Q^{et}.\,\bigcup(X\,(\lambda x^e.\,\wp\,(\lambda v^s.\,Q\,x)))\,w) \\ )& \end{aligned}$$

Table IV.1 – Interpretation $\mathfrak{L}_{\mathbf{q}}$ of linguistic constants

Adding $?_{\mathbf{q}}$ before $\exists_{\mathbf{q}}$ allows answers to be negative, e.g. "Nobody sleeps" or "Marie sleeps nowhere". The question closure for location is however debatable, as an event is often supposed to happen somewhere. Moreover, this interpretation would fail to capture *mention-all readings*, asking for a complete specification for each entity. For example, **Qui dort ?** could also be understood as

requiring, for each individual a, whether a sleeps, that is $\forall x^{e}.?\,(\wp\,(\lambda w^{s}.\,\mathbf{sleep_i}\,w\,x))$ instead of (IV-4-b).

We give final $\beta\eta$-reduced semantic representations in $\Sigma_{\mathbf{q}}$.

Set 1 and especially **loc** remarkably simplifies to applying $\wp$:

(2) a. $\wp\,(\lambda w^{s}.\,\mathbf{sleep_i}\,w\,\mathbf{m})$
b. $\wp\,(\lambda w^{s}.\,\exists z^{\iota}.\,(\mathbf{house_i}\,w\,z)\wedge_{\mathbf{e}}(\mathbf{own_i}\,w\,z\,\mathbf{c})\wedge_{\mathbf{e}}(\mathbf{loc_i}\,w\,(\lambda Y^{(et)t}.\,Y\,(\mathbf{sleep_i}\,w))\,z\,(\lambda Q^{et}.\,Q\,\mathbf{m})))$

Set 2:

(3) a. $\wp\,(\lambda v^{s}.\,\mathbf{M_q}\,(\wp\,(\lambda w.\,\mathbf{sleep_i}\,w\,\mathbf{m}))\,v)$
b. $\wp\,(\lambda v^{s}.\,\mathbf{K_q\,j}\,(\wp\,(\lambda w.\,\mathbf{sleep_i}\,w\,\mathbf{m}))\,v)$

Set 3–4 of simple questions:

(4) a. $?_{\mathbf{q}}\,(\wp\,(\lambda w^{s}.\,\mathbf{sleep_i}\,w\,\mathbf{m}))$
b. $?_{\mathbf{q}}\,(\exists_{\mathbf{q}}\,(\lambda x^{e}.\ \wp\,(\lambda w^{s}.\,\mathbf{sleep_i}\,w\,x)))$
c. $?_{\mathbf{q}}\,(\exists_{\mathbf{q}}\,(\lambda z^{e}.\ \wp\,(\lambda w^{s}.\,\mathbf{loc_i}\,w\,(\lambda X^{E_{\mathbf{e}}}.\,X\,\mathbf{sleep})\,z\,(\lambda Q^{et}.\,Q\,\mathbf{m}))))$

Set 5 of embedded questions:

(5) a. $\mathbf{K_q\,j}\,(?_{\mathbf{q}}\,(\wp\,(\lambda w^{s}.\,\mathbf{sleep_i}\,w\,\mathbf{m})))$
b. $\mathbf{K_q\,j}\,(?_{\mathbf{q}}\,(\exists_{\mathbf{q}}\,(\lambda x^{e}.\ \wp\,(\lambda w^{s}.\,\mathbf{sleep_i}\,w\,x))))$
c. $\mathbf{K_q\,j}\,(?_{\mathbf{q}}\,(\exists_{\mathbf{q}}\,(\lambda z^{e}.\ \wp\,(\lambda w^{s}.\,\mathbf{loc_i}\,w\,(\lambda X^{E_{\mathbf{e}}}.\,X\,\mathbf{sleep})\,z\,(\lambda Q^{et}.\,Q\,\mathbf{m})))))$

V.
Conclusion

Inquisitive semantics improves intentional semantics by producing a uniform semantic representation for interrogative and declarative sentences. It models clauses by a non-empty downward-closed set of states, each state being a set of possible worlds. Questions have several maximal elements called alternatives, corresponding to their possible answers. Assertions have only one alternative and thus represent the meaning of declarative sentences. Therefore, we can transform an intentional meaning S into an inquisitive one by taking its powerset $\wp(S)$.

This dissertation investigates a systematic transformation from lexical intentional meanings to inquisitive ones, which preserves the original logical properties and is compatible with composition. Such a transformation is named a conservative extension.

After a summary of needed mathematical notions about λ-calculus and its semantics in section I, we provided in section II a syntactic analysis of French questions. We focused on declarative questions with *in situ* interrogative pronouns (subject and locative adjunct) and questions with the particle est-ce que, either standing alone or embedded in a clause with a main responsive verb. We built a toy fragment of French and gave a deep syntax analysis with an abstract categorial grammar.

In section III, we provided a Montagovian semantic interface to our deep syntax. We extended this extensional semantics to an intentional semantic lexicon.

The goal of section IV was to design a conservative extension from intentional to inquisitive semantics, called inquisitivation. We briefly presented inquisitive semantics and how we can emulate first-order epistemic inquisitive logic with a class of Henkin models. After recalling de Groote's construction [23], we prove that, under certain monotonicity conditions, this construction also preserve entailment. Finally, we define inquisitivation and explore how it acts on logical connectives and linguistic constants.

The diagram of all vocabularies considered in this dissertation is displayed in Fig. IV.1.

As future prospects, improving the deep syntax to get a second-order ACG and adapting the object language to neo-Davidsonian semantics are planned. This way, a much larger fragment of French could be used to test question syntax modeling and inquisitivation. The semantic part can also be improved by adding presuppositions and a dynamic treatment of anaphors.

The theorem of entailment preservation extends to the other conservative extensions considered in [23]. However, it would be interesting to investigate whether we can weaken the monotonicity conditions to obtain it. Using the more abstract reformulation in category theory begun in [39] could help refine this.

Bibliography

[1] Maria Aloni. Free choice, modals, and imperatives. *Natural Language Semantics*, 15(1):65–94, March 2007.

[2] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Applied Logic Series. Springer Netherlands, second edition, 2002(1986).

[3] Kata Balogh. *Theme with Variations : A Context-Based Analysis of Focus*. PhD thesis, Amsterdam Institute for Logic, Language and Computation, 2009.

[4] Henk Barendregt. *The Lambda Calculus. Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North Holland, second edition, September 2014.

[5] Cedric Boeckx. *Syntactic Islands*. Key Topics in Syntax. Cambridge University Press, Cambridge, 2012.

[6] George Boole. *An Investigation of the Laws of Thought On Which Are Founded the Mathematical Theories of Logic and Probabilities*. Macmillan, 1854.

[7] Maria Boritchev and Philippe de Groote. On dialogue modeling: A dynamic epistemic inquisitive approach. In *LENLS17 : Logic & Engineering of Natural Language Semantics*, November 2020.

[8] Lucas Champollion. The interaction of compositional semantics and event semantics. *Linguistics and Philosophy*, 38(1):31–66, February 2015.

[9] Lisa Lai-Shen Cheng and Johan Rooryck. Licensing Wh-in-situ. *Syntax*, 3(1):1–19, 2000.

[10] Noam Chomsky. *A Minimalist Program for Linguistic Theory*. The MIT Press, 1993.

[11] Ivano Ciardelli. A first-order inquisitive semantics. In Maria Aloni, Harald Bastiaanse, Tikitu de Jager, and Katrin Schulz, editors, *Logic, Language and Meaning: Revised Selected Papers from the 17th Amsterdam Colloquium*, Lecture Notes in Artificial Intelligence, pages 134–143, Berlin Heidelberg, 2010. Springer-Verlag.

[12] Ivano Ciardelli, Jeroen Groenendijk, and Floris Roelofsen. Inquisitive Semantics: A New Notion of Meaning. *Language and Linguistics Compass*, 7(9):459–476, 2013.

[13] Ivano Ciardelli, Jeroen Groenendijk, and Floris Roelofsen. On the semantics and logic of declaratives and interrogatives. *Synthese*, 192(6):1689–1728, June 2015.

[14] Ivano Ciardelli, Jeroen Groenendijk, and Floris Roelofsen. *Inquisitive Semantics*. Oxford Surveys in Semantics and Pragmatics. Oxford University Press, Oxford, New York, November 2018.

[15] Ivano Ciardelli and Floris Roelofsen. Inquisitive Logic. *Journal of Philosophical Logic*, 40(1):55–94, February 2011.

[16] Ivano Ciardelli and Floris Roelofsen. Inquisitive dynamic epistemic logic. *Synthese*, 192(6):1643–1687, June 2015.

[17] Ivano Ciardelli, Floris Roelofsen, and Nadine Theiler. Composing alternatives. *Linguistics and Philosophy*, 40(1):1–36, February 2017.

[18] Robin Cooper. Records and Record Types in Semantic Theory. *Journal of Logic and Computation*, 15(2):99–112, April 2005.

[19] Thierry Coquand. Course Notes in Typed Lambda Calculus. 2008.

[20] Jacques Damourette and Édouard Pichon. *Essai de grammaire de la langue française : des mots à la pensée. Tome 2. 19100 - 1930 : adjectif nominal, adverbe, interjection, phrase*. Paris, edition d'artrey edition, 1968.

[21] Donald Davidson. Truth and Meaning. *Synthese*, 17(1):304–323, 1967.

[22] Philippe de Groote. Tree-Adjoining Grammars as Abstract Categorial Grammars. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6)*, pages 145–150, Universitá di Venezia, May 2002. Association for Computational Linguistics.

[23] Philippe de Groote. On Logical Relations and Conservativity. In *EPiC Series in Computing*, volume 32, pages 1–11. EasyChair, July 2015.

[24] Philippe de Groote. Lambek Categorial Grammars as Abstract Categorial Grammars. In *LENLS 13*, October 2016.

[25] Philippe de Groote and Makoto Kanazawa. A Note on Intensionalization. *Journal of Logic, Language and Information*, 22(2):173–194, April 2013.

[26] Philippe De Groote and Sarah Maarek. Type-theoretic extensions of abstract categorial grammars. November 2007.

[27] Philippe de Groote, Sarah Maarek, and Ryo Yoshinaka. On Two Extensions of Abstract Categorial Grammars. In Nachum Dershowitz and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, Lecture Notes in Computer Science, pages 273–287, Berlin, Heidelberg, 2007. Springer.

[28] Philippe de Groote and Sylvain Pogodalla. On the Expressive Power of Abstract Categorial Grammars: Representing Context-Free Formalisms. *Journal of Logic, Language and Information*, 13:421–438, March 2004.

[29] Philippe de Groote and Yoad Winter. *A Type-Logical Account of Quantification in Event Semantics*. November 2014.

[30] Samuel Eilenberg and G. Max Kelly. Closed Categories. In S. Eilenberg, D. K. Harrison, S. MacLane, and H. Röhrl, editors, *Proceedings of the Conference on Categorical Algebra*, pages 421–562, Berlin, Heidelberg, 1966. Springer.

[31] Lucien Foulet. Comment ont évolué les formes de l'interrogation. *Romania*, 47(186):243–348, 1921.

[32] Jean Goubault-Larrecq. Aspects logiques. 2021.

[33] Jean Goubault-Larrecq. Lambda-calcul et langages fonctionnels. 2021.

[34] Charles Leonard Hamblin. Questions in Montague English. *Foundations of Language*, 10(1):41–53, 1973.

[35] Irene Heim and Angelika Kratzer. *Semantics in Generative Grammar*. Blackwell, 1998.

[36] Leon Henkin. Completeness in the Theory of Types. *The Journal of Symbolic Logic*, 15(2):81–91, 1950.

[37] J. Roger Hindley. *Basic Simple Type Theory*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, 1997.

[38] Jaakko Hintikka. Knowledge and Belief: An Introduction to the Logic of the Two Notions. *Studia Logica*, 16:119–122, 1962.

[39] Mathieu Huot. *Conservative Extensions of Montague Semantics*. Master thesis, ENS Cachan, Université Paris-Saclay, 2017.

[40] Makoto Kanazawa. A Prefix-Correct Earley Recognizer for Multiple Context-Free Grammars. In *Proceedings of the Ninth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+9)*, pages 49–56, Tübingen, Germany, June 2008. Association for Computational Linguistics.

[41] Angelika Kratzer and Junko Shimoyama. Indeterminate Pronouns: The View from Japanese. *undefined*, 2002.

[42] Saul A. Kripke. A Completeness Theorem in Modal Logic. *The Journal of Symbolic Logic*, 24(1):1–14, 1959.

[43] Utpal Lahiri. *Questions and Answers in Embedded Contexts*. OUP Oxford, Oxford ; New York, January 2002.

[44] Ekaterina Lebedeva. *Expressing Discourse Dynamics Through Continuations*. PhD thesis, Université de Lorraine, April 2012.

[45] Jiri Marsik. *Towards a Wide-Coverage Grammar : Graphical Abstract Categorical Grammars*. Other, Université de Lorraine, June 2013.

[46] Richard Montague. *English as a Formal Language*. De Gruyter Mouton, 1970.

[47] Richard Montague. Universal grammar. *Theoria*, 36(3):373–398, 1970.

[48] Richard Montague. The Proper Treatment of Quantification in Ordinary English. In K. J. J. Hintikka, J. M. E. Moravcsik, and P. Suppes, editors, *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*, Synthese Library, pages 221–242. Springer Netherlands, Dordrecht, 1973.

[49] Richard Moot. Hybrid type-logical grammars, first-order linear logic and the descriptive inadequacy of lambda grammars. Research report, arXiv, 2014.

[50] Annick Morin. *Questioning Particles: A Cross-Linguistic Approach to Quebec French Polar Interrogatives*. Thesis, June 2017.

[51] Stefan Müller, Anne Abeillé, Robert D. Borley, and Jean-Pierre Koenig. *Head-Driven Phrase Structure Grammar, the Handbook*. Empirically Oriented Theoretical Morphology and Syntax. Language Science Press, Berlin, October 2019.

[52] Sylvain Pogodalla. ACGtk : un outil de développement et de test pour les grammaires catégorielles abstraites. In *Actes de la conférence conjointe JEP-TALN-RECITAL 2016. volume 5 : Démonstrations*, pages 1–2, Paris, France, 2016. AFCP - ATALA.

[53] Sylvain Pogodalla and Florent Pompigne. Controlling Extraction in Abstract Categorial Grammars. In *FG*, 2010.

[54] Valentin D. Richard. Traduction des grammaires catégorielles de lambek dans les grammaires catégorielles abstraites. 2018.

[55] Mats Edwards Rooth. *Association with Focus*. Phd thesis, University of Massachusetts, January 1985.

[56] Sylvain Salvati. Encoding second order string ACG with Deterministic Tree Walking Transducers. In Shuly Wintner, editor, *The 11th Conference on Formal Grammar*, FG Online Proceedings, pages 143–156, Malaga, Spain, 2006. CSLI Publications.

[57] Sylvain Salvati. A Note on the Complexity of Abstract Categorial Grammars. In Christian Ebert, Gerhard Jäger, and Jens Michaelis, editors, *The Mathematics of Language*, Lecture Notes in Computer Science, pages 266–271, Berlin, Heidelberg, 2010. Springer.

[58] Félix Tanguay. D'où vient le «-tu» interrogatif, et «c'est-tu» pertinent de l'enseigner? In *Histoire de la grammaire*, volume 25. Correspondance, 2020.

[59] Nadine Theiler. *A Multitude of Answers: Embedded Questions in Typed Inquisitive Semantics*. MSc. Thesis, August 2014.

[60] Yoad Winter and Joost Zwarts. Event Semantics and Abstract Categorial Grammar. In Makoto Kanazawa, András Kornai, Marcus Kracht, and Hiroyuki Seki, editors, *The Mathematics of Language*, Lecture Notes in Computer Science, pages 174–191, Berlin, Heidelberg, 2011. Springer.

[61] Andrzej Wiśniewski. The Posing of Questions: Logical Foundations of Erotetic Inferences. *Studia Logica*, 61(2):296–299, 1995.

[62] Edward N. Zalta. Gottlob Frege. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2020 edition, 2020.

Index

α-equivalence, 11
β-reduction, 15
η-expansion, 15
λ-term, 10
    linear λ-term, 12

abstract categorial grammar (ACG), 22
alternative, 42, 45
assertive, 45
assignment, 16, 31

bound variable, 11

characteristic function, 31
closed, 11
complete Boolean algebra, 34
connective, 30
conservative extension, 40, 49

deduction system, 7
definition domain, 7
derivable, 13
derivation, 7

embedding, 49, 52
entailment, 18, 32
entity, **see** individual
extraction, 20
extraction island, 20

first-order logic (FOL), 31
FOEIL : first-order epistemic inquisitive logic, 43
formula, 31
free variable, 11
fresh variable, 11

ground term, 9

Henkin model, 16
higher-order signature (HSO), 10

individual, 30, 33
information state, 44
inquisitivation, 52
inquisitive semantics, 43
intentional logic, 38
issue, 44

lexicon, 22
logical equivalence, 19

meta-variable, 7
model, 31
monotonic, 19

noun phrase, 26

order, 23

partial map, 7
powerset $\wp$, 43
predicate, 30
projection, 49, 52
proposition, 30, 33

question, 45

rule, 7

substitution, 11

type
    linear type, 12
    simple type, 10
typing derivation, 12

union set $\bigcup$, 44

variable
    λ-variables, 10
    type variable, 9
vocabulary, 22
    abstract vocabulary, 22
    object vocabulary, 22