

Eleni Kalyvianaki

ALGORITHMIC NATURAL LANGUAGE SEMANTICS

A Study of Locality in the Theory of Referential Intensions

DOCTORAL DISSERTATION



September 2007

Graduate Program in Logic, Algorithms and Computation (μΠλΥ)
Department of Mathematics
National and Kapodistrian University of Athens

The dictionaries know all the stories
that we write and that we will write
in the future, they have just forgotten them.
Our role is to remind these stories
to them.

Vassilis Alexakis

Acknowledgments

I would like to thank wholeheartedly my supervisor, Prof. Y.N. Moschovakis, for the endless hours he has devoted in this work, his never ending patience towards my occasional stubbornness and his constant encouragement and persistence. Besides his exemplary mathematical teaching, he was an example of a “way of life” which I consider as the most important “lesson” of my studies.

I have to express my gratitude to the Graduate Program in Logic, Algorithms and Computation ($\mu\Pi\lambda\forall$) for the studies I had the opportunity to pursue. I warmly thank all the academic and administrative staff that supports its operation during the last years.

There have been a lot of people who helped me in the course of my studies and I thank them all. I thank Prof. Joan Rand Moschovakis for her warm hospitality but also her encouraging and apt remarks on various occasions. I would also have to thank Prof. Yannis Stephanou for his eagerness to respond to all my questions and also Prof. Fritz Hamm and Prof. David Kaplan for their time. I also thank the Department of Mathematics of U.C.L.A. for its hospitality during April-May 2005. Finally, I thank for their perfect cooperation the members of the Three Member Committee: Prof. C. Dimitrakopoulos and Prof. G. Koletsos and the other members of the Seven Member Exam Committee: Prof. L. Kirousis, Prof. Ant. Melas, Prof. Y. Stephanou, Prof. Ath. Tzouvaras and Prof. St. Psillos.

My graduate studies was co-funded by the European Union - European Social Fund and National Resources - EPEAEK II.

Pursuing a Ph.D. diploma is an adventure into which the close friends and the members of the family of the candidate participate whether they like it or not. This adventure has its ups and downs for the candidate herself but for them, it only means anxiety and the need to show never ending patience. Someone told me once that till the end of one’s graduate studies, he “has lost” friends and family; I think that this hasn’t happened to me.

I thus have to thank my parents for never questioning my decisions and

for supporting me with all their heart, even contrary to their own ideas some times. During all these years, my sister stood always by me, listening patiently to my complaints and anxieties during endless hours of telephone calls from dull Cambridge. I wish her with all my heart to finish soon her own studies as well.

I cannot also fail to acknowledge the patience and support in many ways of my boyfriend N., who accompanied me in this adventure, hoping that I will have the generosity to compensate him in the future.

Finally, I thank a loyal “friend” who was always ready to listen to me “with the outmost attention” to explain parts of this dissertation.

Contents

Introduction	1
1 Formal Models of Meaning	5
1.1 Gottlob Frege: The Origins	5
1.2 The Typed λ -Calculus	8
1.3 Richard Montague: Formal Semantics	10
1.3.1 LIL: Language of Intensional Logic	12
1.3.2 Natural Language Examples in LIL	14
1.4 David Kaplan: Logic of Demonstratives	16
1.5 Theory of Referential Intensions	18
1.5.1 Syntax and Semantics of L_{ar}^λ	18
1.5.2 Referential Synonymy and Local Meaning	23
1.6 Two Notions of Situated Meaning	25
2 Locality of Typed Objects	31
2.1 State-dependent Types of L_{ar}^λ	31
2.2 Local Objects	33
2.3 Locality Indices of Type $\tilde{\sigma}$	39
2.4 Associate with respect to an Input Index	43
3 Locality of Terms	55
3.1 Local Terms	55
3.2 Closed Locality Proofs	57
3.3 The Most Local Locality Proof	66
4 Formal Associates of Terms	79
4.1 An Extension of L_{ar}^λ	79
4.2 Formal Local Associates of Local Terms	82
4.3 Formal Associates of General Terms	87

5	Algorithmic Factual Content	97
5.1	Factual Canonical Form and Factual Synonymy	98
5.2	Factual Content and the Logic of Demonstratives	103
5.3	Future Work	105
	Bibliography	107
	Index	111

Introduction

As with any other system of signs, we use natural language expressions (the signs) to “talk about”, to *refer* to objects of the world around us. We communicate by means of a language if we know the way this mapping is performed. Gottlob Frege (Section 1.1) argued that language expressions, apart from their *reference*, have also *meanings* that determine their references but are not exhausted by them.

Formal semantics, as conceived by Richard Montague (Section 1.3), seek to study natural language semantics in exactly the same way as one studies the semantics of any formal language in Mathematical Logic. The results are truth-conditional, model-theoretic theories of meaning where for each term of the language both meaning and reference are defined formally.

Context of reference is an indispensable part of language usage. In oral or written form, we usually interpret and thus, understand language with respect to a context that permits us to attribute specific references to language expressions. For example, the *indexical* ‘I’ refers each time to the person who utters the sentence in which it occurs. In formal theories of meaning, the reference of a term is defined at a particular context while its meaning determines the corresponding reference at every context.

Another characteristic of these theories is that they adhere to the *Compositionality Principle*: “*The meaning of a whole is a function of the meanings of its parts and their mode of syntactic combination*” (as stated in the Introduction of [24]). In this way, we explain how a language speaker can understand a new sentence that she has never heard before or how a speaker augments her knowledge of the language.

The work presented here is developed within the *Theory of Referential Intentions* (Section 1.5). It is a formal theory of meaning that is part of the Fregean tradition but at the same time, it introduces a whole new dimension in formal semantics. The key idea is that ([22])

...the meaning of a term A can be faithfully modeled by its ref-

referential intension $\text{int}(A)$, an (abstract, idealized, not necessarily implementable) algorithm which computes the denotation [reference] of A .

The meaning of a sentence in this theory determines the parts that we use to compute its denotation, codifies the way this computation is done and it is through this process that it determines its denotation. This structural notion of meaning yields a calculus of *referential synonymy* which makes fine distinctions between corresponding language expressions that go beyond their references or even, their references at any particular context. In the Theory of Referential Intensions, we attribute two kinds of meaning to a sentence A : its *global meaning*, that is, the algorithm that computes the denotation of A (Truth or Falsity) at every possible context of reference and its *local meaning* at a context which is the algorithm that computes the denotation of A at that particular context.

The problem that we address in this work is to investigate a notion of situated, structural meaning of a sentence at a context of reference which computes the denotation at a context directly from the corresponding, situated meanings of the parts of the sentence that are needed for the computation. The aim is to define a semantical value for each sentence at a context of reference, its **factual content**, which expresses formally **what the sentence says about the world at that particular context**. Thus, at a context of reference where ‘John’ and ‘He’ refer to the same person, the sentences ‘John runs’ and ‘He runs’ should have the same factual content although they are not locally synonymous; on the other hand, if ‘the brother of Mary’ refers to that particular person also, the factual content of the sentence ‘The brother of Mary runs’ differs since it expresses additional facts about the world.

A key idea in the approach adopted here, was to define formally and study a notion of **locality** that explains the way situated meanings of the parts of a sentence are combined. The interpretation of each constant of the language determines whether its value at a particular context depends only on the values of its arguments at that context or on their values at other contexts as well. For example, compare the verb ‘run’ and the sentential operator ‘necessarily’. The goal was then to produce a *most local* interpretation of each sentence that respects the locality behavior of the constants that occur in it but in the same time, the computation of the denotation of any of its subterms at a context is restricted on the values of its arguments at that context *as much as possible*. So ultimately, from the mathematical point of view, this is a study of locality in the Theory of Referential Intensions.

The thesis is organized in five chapters as follows. In Chapter 1, we

present in detail certain formal theories of meaning that follow the inaugural ideas of G. Frege, focusing on their treatment of situated meaning. We also introduce the formal language \mathcal{L}_{ar}^λ , the language of the Theory of Referential Intentions, which is the technical framework into which this work is developed. In Chapter 2, the fundamental ideas of locality are introduced for objects in the interpretation structure of \mathcal{L}_{ar}^λ while in Chapter 3, these ideas are developed into a study of the locality behavior of terms. In this latter chapter, we also show that there exists a most local interpretation for each term. In Chapter 4, we define formally in \mathcal{L}_{ar}^λ for every term A a new term that reflects the locality behavior of A . Finally, in Chapter 5, for each term A , we use this new term under its most local interpretation to define the factual content of A at every context of reference.

Thus, our main object of study is not introduced formally until the last chapter of the thesis. Actually, although factual content is intuitively a clear notion for simple examples in natural language, its precise definition turned out to be fairly complex, especially since it was studied here under a general notion of locality.

Chapter 1

Formal Models of Meaning

In the tradition of formal semantics, meaning in natural language is modeled by various mathematical notions in an attempt to define rigorously Fregean *sense* and *denotation*. In this chapter, we present the notions of global and local meaning as they are defined in the work of R. Montague (Section 1.3) and of D. Kaplan (Section 1.4) and in the Theory of Referential Intentions (Section 1.5). The last part is the technical framework of this thesis and thus, it is introduced in a more detailed way.

In Section 1.2, there is a short account of the typed λ -calculus which is the basic tool of the theories presented here while in Section 1.6, we comment on the basic results of [13] where two notions of situated meaning — local meaning and factual content — are defined in addition to a notion of global meaning in an “algorithmic” extension of Montague’s logic. Throughout the chapter, attention is paid in the different ways that situated meaning is defined and treated.

The chapter begins with a short account of the ideas of G. Frege — where it all began.

1.1 Gottlob Frege: The Origins

In [6], G. Frege treats natural language as a system of signs and seeks to understand it as he would do with any other such formal system with the means of logical analysis. This kind of abstraction towards natural language is not straightforward but it surely comes as no surprise to someone whose primary interest is the logical foundations of mathematics.

First and foremost, the “signs” of natural language are used by speakers to *refer* to the world in its full complexity and totality. Frege argues, though,

that apart from the reference of each sign (its *denotation*, *Bedeutung*)¹, there is also its *sense* (*Sinn*), “wherein the mode of presentation is contained” ([6]: p. 57)². Thus, to mention a classical example, the two phrases ‘the evening star’ and ‘the morning star’ have the same denotation; they both refer to the planet Venus. Nevertheless, the two phrases do not have the same sense, which explains why the statement ‘the morning star is the evening star’ says something non trivial about our world.

A language speaker knows the senses of the “names” or signs and by their mediation may be able to determine their denotations (if they exist). Although senses are not formally defined³, Frege gives emphasis to their objective role in contrast to the subjective one of “ideas”. To sum up, in one phrase: “a proper name (...) expresses its sense, stands for or designates its reference” ([6]: p. 61).

The next step in Frege’s consideration of natural language semantics is the *declarative sentences* (sentences, for short). First of all, Frege considers them as being composed of parts much like mathematical expressions can be decomposed into a function and its arguments. In his own words ([3]: p. 31)

Statements in general, just like equations or inequalities of expressions in Analysis, can be imagined to be split up into two parts; one complete in itself, and the other in need of supplementation, or ‘unsaturated’. Thus, e.g., we split up the sentence

‘Caesar conquered Gaul’

into ‘Caesar’ and ‘conquered Gaul’. The second part is ‘unsaturated’ — it contains an empty place; only when this place is filled up with a proper name, or with an expression that replaces a proper name, does a complete sense appear. Here too I give the name ‘function’ to what this ‘unsaturated’ part stands for. In this case the argument is Caesar.

Secondly, a sentence as a whole, much like a “name”, has a sense and a denotation. Its sense is a “thought”: “(...) not the subjective performance of thinking but its objective content, which is capable of being the common

¹In [6], “Bedeutung” is translated as “meaning” but we will use here the term “denotation” instead, leaving the term “meaning” for more general uses.

²In all the extracts included in this chapter, the page numbers refer to the corresponding reference in English.

³It is maybe because “what is logically simple cannot have a proper definition” ([5], p. 193).

property of several thinkers” ([6]: p. 62). Its denotation is a truth value (Truth or Falsity) which together with its sense yields knowledge.

The constituent parts of any sentence contribute by their denotations to the denotation of the sentence. Frege states explicitly that when a part of a sentence is replaced by a phrase with the same denotation, then the denotation of the sentence remains the same although its sense may change.

There is a lively controversy (cf. [11], [23]) whether he believes an analogous *substitution property* for senses and whether this property, which is clearly stated for denotations, implies the Compositionality Principle for denotations or senses, especially in relation to his *Context Principle*⁴ ([4]), which, however, was formulated much earlier. It is fair to say that the substitution property for senses is not explicitly formulated in [6] but there are extracts where it is, in a way, implied; for example, ([6]: p. 62-63)

If it were a question only of the sense of the sentence, the thought, it would be unnecessary to bother with the reference of a part of the sentence; only the sense, not the reference, of the part is relevant to the sense of the whole sentence.

One thing that is worth mentioning here is that substitution of a part of a sentence by an other expression with *the same sense* presupposes that some kind of equivalence relation between senses must be defined and used. This is not possible for Frege since senses are not formally defined, although he states that the same sense can be expressed in various ways and thus the possibility exists ([5]: p. 46).

What is the role of the context of reference in Fregean semantics? It is not straightforward to say and one has to be cautious not to try to find in Frege answers to questions that were posed several decades after his work. Nevertheless, there is a clear assumption in Frege that a sentence is always considered with respect to a context of reference — usually the current time, place, etc. The denotation of a sentence is Truth or Falsity exactly because the context of reference is always implicit in the use of natural language. A passage in [7] ([25]: p. 40), although written much later than [6], is illuminating:

Therefore the time of utterance is part of the expression of the thought. If someone wants to say today what he expressed yesterday using the word ‘today’, he will replace this word with

⁴The Context Principle, characterized thus by Frege scholars and not Frege himself, states that the reference (denotation, Bedeutung) of words is to be determined in the context of a sentence and not in isolation.

‘yesterday’. Although the thought is the same, its verbal expression must be different in order that the change of sense which would otherwise be effected by the differing times of utterance may be canceled out.

Thus, according to Frege, an utterance of ‘It is raining today’ at a particular day expresses the same thought (the same sense) with the utterance of ‘It was raining yesterday’ the next day. They have, of course, the same denotation but, in addition, they express the same thought. Frege points here at a *synonymy relation between two semantic values* that is intuitively appealing and close to our experience.

Another example that is mentioned in [7] is that of an utterance involving the indexical ‘I’. It is compared with a corresponding utterance where in the place of ‘I’ the name of the speaker is mentioned, namely ‘Dr Gustav Lauben’. In this case the synonymy relation is not at all clear because both the sense of ‘I’ and the sense of a proper name are not trivially determined and they involve the speaker’s perception of them.

To sum up, in Fregean semantics, the denotation and the sense of a sentence are always considered with respect to a context of reference. His interest is primarily in situated meaning since Truth or Falsity of a sentence can only be determined if we consider it at a particular context and to express the same sense in two different contexts may require the utterance of two distinct sentences.

1.2 The Typed λ -Calculus

The beginnings of the λ -calculus in the 30’s by Alonzo Church were a quest of foundation of mathematics. It is used as a general framework for the study of functions where a function is not exhausted by the set of pairs of argument and value but rather is “expressed” by the way these pairs are produced. In this section, the λ -calculus is introduced briefly in the form that it is used in both the work of R. Montague and in the Theory of Referential Intentions.

Types are defined recursively by a set of basic types and the function type over them. In our consideration here the basic types are: entities (**e**), truth values (**t**) and, possibly, states (**s**), and, in general

$$\tau ::= \mathbf{e} \mid \mathbf{t} \mid \mathbf{s} \mid (\tau_1 \rightarrow \tau_2). \quad (1.1)$$

We assume that we have an infinite number of typed variables for each type. If x is a variable of type τ , we use the notation x^τ . *Terms* are defined

using two operations, *application* and *λ -abstraction* by the recursion

$$A ::= x \mid A(B) \mid \lambda(x)(B). \quad (1.2)$$

Each term is assigned a type noted as

$$A : \tau \iff \text{the type of } A \text{ is } \tau$$

by the following type restrictions:

- If x is of type τ (x^τ), then as a term $x : \tau$.
- If $A : (\tau_1 \rightarrow \tau_2)$ and $B : \tau_1$, then $A(B) : \tau_2$.
- If $B : \tau_2$ and $x : \tau_1$, then $\lambda(x)(B) : (\tau_1 \rightarrow \tau_2)$.

The *free* occurrences of a variable x in a term A are defined by⁵:

- If $A \equiv x$, then the occurrence of x is free in A .
- If $A \equiv B(C)$, then the free occurrences of x in A are those of x in B and C .
- If $A \equiv \lambda(x)(B)$, no occurrence of x is free in A but the free occurrences of any $y \neq x$ in A are those of y in B .

In the case of the λ -term, $A \equiv \lambda(x)(B)$, every occurrence of x in A is considered *bound*. And a term is *closed* if there are no free occurrences of variables in it.

The formal systems that we will consider are enriched by typed *constants* (noted usually by $c : \tau$) and thus, terms are in general defined by the recursion

$$A ::= x \mid c \mid A(B) \mid \lambda(x)(B). \quad (1.3)$$

For every term A , we can also define its *formation tree* which is a suitable representation of the way it is formed.

A standard structure of a typed λ -calculus system comprises:

- (i) Non empty sets (*universes*) of objects of the basic types (\mathbb{T}_τ) and for the function types, the corresponding sets of functions as follows:

$$\mathbb{T}_{\tau_1 \rightarrow \tau_2} = \{f \mid f : \mathbb{T}_{\tau_1} \rightarrow \mathbb{T}_{\tau_2}\}.$$

⁵We will generally use the symbol “ \equiv ” to denote the identity relation on syntactic objects.

(ii) If there are constants, for each constant $c : \tau$, a corresponding object of the same type $c \in \mathbb{T}_\tau$.

An *assignment* function g associates to each variable $x : \tau$ some object in the corresponding universe of objects of type τ ($g(x) \in \mathbb{T}_\tau$). We also define the *update* $g\{x := f\}$ of an assignment g by

$$g\{x := f\}(y) = \begin{cases} f, & \text{if } y \equiv x, \\ g(y), & \text{otherwise.} \end{cases}$$

Finally, a denotation function is defined in this structure such that each term $A : \tau$ and each assignment g to the variables is associated with an object in the corresponding \mathbb{T}_τ . We define the function

$$\text{den}(A) : \text{Assignments} \rightarrow \mathbb{T}_\tau$$

by the following recursive clauses:

- $\text{den}(x)(g) = g(x)$.
- $\text{den}(c)(g) = c$.
- $\text{den}(B(C))(g) = \text{den}(B)(g)(\text{den}(C)(g))$.
- $\text{den}(\lambda(x)(B))(g) = (f \mapsto \text{den}(B)(g\{x := f\}))$,
where if $x : \tau$, then $f \in \mathbb{T}_\tau$.

1.3 Richard Montague: Formal Semantics

R. Montague's work in the 60's founded *formal natural language semantics*; it was motivated by his clearly stated conviction that opens [16]:

I reject the contention that an important theoretical difference exists between formal and natural languages.

Montague develops and defends his conviction in a series of papers (collected in [19]) that aim to provide English language with semantics much like the semantics of any formal language of symbolic logic.

Natural language is translated into a formal language (*Intensional Logic* – **IL**) by a detailed procedure which maps its syntactic rules into appropriate term formation rules of **IL**. Thus, any phrase, any sentence is *rendered* by a term in **IL** which is a typed λ -calculus equipped with an *intension* ($\hat{}$) and an *extension* ($\check{}$) operator. Interpretation in **IL** depends on context of

reference, and so every term A denotes its *extension* at any given context; the *intension* of A is the function which assigns to each context the extension of A at that context and it is denoted in \mathbb{L} by the term \hat{A} .

Before presenting in detail a version of Montague's formal system, we will make some general comments on the background of this work and its connections with Frege's ideas. Rudolf Carnap was the first to model Frege's sense and denotation by intension and extension, respectively, and to formalize the notion of context of reference as a *state-description* ([1]: p. 9):

*A class of sentence in S_I , which contains for every atomic sentence either this sentence or its negation, but not both, and no other sentences, is called a **state-description** in S_I , because it obviously gives a complete description of a possible state of the universe of individuals with respect to all properties and relations expressed by predicates of the system. Thus the state-descriptions represent Leibniz' possible worlds or Wittgenstein's possible states of affairs.*

The extension of an expression is now its denotation at a particular state-description and its intension is a function from all such state-descriptions to extensions — at every possible such state, the intension will determine the corresponding denotation.

Montague's \mathbb{L} is based on these ideas but also builds on Saul Kripke's semantics for Modal Logic where *possible world semantics* are defined formally in a concrete way. *States* or *indices*, as used by Montague, and by all the other formal semantics systems (with many but not essential variations), are tuples of indicators that stand for each of the multiple aspects of a context of reference. Usually, these indicators include but are not exhausted by: a possible world, a particular moment of time, a particular place, an agent who is the speaker etc. In any case, it is a basic presupposition in such systems (as in the ones that we will present and use here) that given a state, all aspects of the context of the reference of an utterance of a particular expression are specified.

Finally, it is an important aspect of Montague's system that it fully respects the Compositionality Principle. In \mathbb{L} , the intension of a complex term is a function of the intensions of its parts and of their syntactic combination.

On the other hand, as Frege argued, there are cases where the denotation of a sentence requires for its determination, not the denotation, but, the sense of a subordinate clause, for example, in *belief sentences* like 'Copernicus believed that the planetary orbits are circles'. In other words, part of the extension of such a sentence is the intension of its subordinate clause. Thus,

in order to adhere to the Compositionality Principle, the intension of an expression is not enough to be part of the *metalanguage* but in contrast, one must be able to refer to it in the *object-language* as well⁶. In the case of IL, the intension and extension operators are the mechanisms that are used to define the corresponding semantic values formally within it.

1.3.1 LIL: Language of Intensional Logic

The *Language of Intensional Logic* (LIL) is a version of IL that is closer to our contemporary understanding of Montague semantics and at the same time lies at the core of L_{ar}^λ which is the formal language of the Theory of Referential Intentions (Section 1.5)⁷.

Types in LIL are defined recursively by

$$\tau ::= e \mid t \mid (s \rightarrow \tau_2) \mid (\tau_1 \rightarrow \tau_2) \quad (1.4)$$

where the *basic* types are: entities (e), truth values (t) and states (s). Notice that s can only occur as the name of the domain of function types⁸.

We assume that there are infinitely many *variables* of every type τ (noted as x^τ). The *terms* in LIL are the λ -calculus terms (Section 1.2) along with the terms formed by the intension and the extension operators, that is, they are defined by

$$A ::= x \mid c \mid A(B) \mid \lambda(x)(B) \mid \sim(A) \mid \hat{}(A) \quad (1.5)$$

where x is a variable of any type, c is a typed constant and some type restrictions must be obeyed. Each term is assigned a type as in the typed λ -calculus:

- If x^τ , then as a term $x : \tau$ and if c is of type τ , then $c : \tau$.
- If $A : (\tau_1 \rightarrow \tau_2)$ and $B : \tau_1$, then $A(B) : \tau_2$.
- If $B : \tau_2$ and $x : \tau_1$, then $\lambda(x)(B) : (\tau_1 \rightarrow \tau_2)$.
- If $A : (s \rightarrow \tau)$, then $\sim(A) : \tau$.
- If $A : \tau$, then $\hat{}(A) : (s \rightarrow \tau)$.

⁶The distinction between object-language and metalanguage is due to R. Carnap in [1].

⁷LIL is also used in [13] (see Section 1.6).

⁸In [8], Gallin argues that s is not a type since “...IL was intended as a formal logic with intensional features close to those of natural languages, and in natural language we do not refer explicitly to contexts of use;”. Dana Scott in [26] argues in the same spirit that “...the answer is that we just do not speak that way.”.

Free and *bound* occurrences of variables in each term are defined as usual.

An interpretation structure of LIL comprises:

- (i) Universes of objects of each type: a non empty set \mathbb{T}_e for entities, the two membered set of truth values $\mathbb{T}_t = \{0, 1\}$ and by recursion for each type $(\tau_1 \rightarrow \tau_2)$,

$$\mathbb{T}_{\tau_1 \rightarrow \tau_2} = \{f \mid f : \mathbb{T}_{\tau_1} \rightarrow \mathbb{T}_{\tau_2}\}.$$

State variables vary over a non empty set \mathbb{T}_s and the set $\mathbb{T}_{s \rightarrow \tau}$ is defined as above.

- (ii) For each constant $c : \tau$, a corresponding function $c : \mathbb{T}_s \rightarrow \mathbb{T}_\tau$.

Now, in an interpretation structure of LIL, the function den_{LIL} associates with each term $A : \tau$ the object

$$\text{den}_{\text{LIL}}(A) : \text{Assignments} \rightarrow (\mathbb{T}_s \rightarrow \mathbb{T}_\tau)$$

and it is defined by the following recursive rules where a and b are states in \mathbb{T}_s :

- $\text{den}_{\text{LIL}}(x)(g)(a) = g(x).$
- $\text{den}_{\text{LIL}}(c)(g)(a) = c(a).$
- $\text{den}_{\text{LIL}}(A(B))(g)(a) = \left(\text{den}_{\text{LIL}}(A)(g)(a)\right)\left(\text{den}_{\text{LIL}}(B)(g)(a)\right).$
- $\text{den}_{\text{LIL}}(\lambda(x)(B))(g)(a) = (f \mapsto \text{den}_{\text{LIL}}(B)(g\{x := f\})(a)),$
where if $x : \sigma$, then f is any object in \mathbb{T}_σ .
- $\text{den}_{\text{LIL}}(\sim(A))(g)(a) = \left(\text{den}_{\text{LIL}}(A)(g)(a)\right)(a).$
- $\text{den}_{\text{LIL}}(\hat{\sim}(A))(g)(a) = (b \mapsto \text{den}_{\text{LIL}}(A)(g)(b)) \quad (= \text{den}_{\text{LIL}}(A)(g)).$

IL syntax and semantics as they are presented in [2] (pp. 155-162) are in complete correspondence with LIL as it is presented here. The terms defined in (1.5) are simply the typed *meaningful expressions* of IL. In IL semantics, the universe of the objects of type τ is the set of *Montague denotations* D_τ which are defined as above while the denotations of type $s \rightarrow \tau$ for any type τ are the *Montague senses of type* τ . Also, to each non-logical constant of type τ is assigned a Montague sense of type τ . Accordingly, an assignment function assigns to each variable of type τ a member of the Montague denotation D_τ .

Names, indexicals	Mary, I, she : e
Common nouns	man, book : e \rightarrow t
Extensional intransitive verbs	run : e \rightarrow t
Extensional transitive verbs	love, read : e \rightarrow (e \rightarrow t)
Articles	the : (e \rightarrow t) \rightarrow e
Quantifiers	every : (e \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t)
(Basic) necessity operator	\Box : (s \rightarrow t) \rightarrow t

Table 1.1: Constants in LIL.

If we now consider each state as being a pair of a possible world and a moment of time, $\langle W, T \rangle$, then it is straightforward that for every term A and any $a \in \mathbb{T}_s$:

$$\text{Montague Extension of } A \text{ at } a : [[A]]^{M,a,g} = \text{den}_{\text{LIL}}(A)(g)(a)$$

$$\text{Montague Intension of } A : [[A]]^{M,g} = \text{den}_{\text{LIL}}(A)(g).$$

Finally, notice that for any term A and a state a ,

$$\text{Montague Extension } (\hat{}(A)) \text{ at state } a = \text{Montague Intension}(A).$$

1.3.2 Natural Language Examples in LIL

By *rendering*, we describe, in general, the procedure of the “translation” of natural language expressions into a formal language such as LIL.

The first step is always an introduction of a set of constants that “represent” words of natural language — they constitute the “formal lexicon” with respect to which we render natural language expressions. In Table 1.1, there are some such constants like *run* that renders the verb ‘run’ or *man* that renders the noun ‘man’. As any other term in LIL, constants are typed. Constants that fall under the same syntactical category are attributed the same type (all intransitive verbs are of the type (e \rightarrow t)) but the same type can be used for two different syntactical categories e.g., the type (e \rightarrow t) is used for both common nouns and intransitive verbs. In general, we adopt here the typing of [13] (Section 1.6) which doesn’t follow Montague faithfully ([18]) but is quite close to [2] (see also the discussion in Section 1.6, page 26).

Montague’s rendering process of natural language into LIL is very precise and rule specific: each syntactic rule corresponds to a translation rule which

determines an appropriate term formation rule. Each expression according to its syntactical category is translated into a term of a specific type. Thus, by the translation rules, language expressions are rendered into terms of LIL that, more or less, represent the way they are understood semantically. Following Frege, functional application is used in most of the cases and there are later approaches where its use is exclusive (see [10] for a relative discussion).

Consider the following two simple examples of natural language sentences and their corresponding renderings.

$$\begin{aligned} \text{Mary runs} &\xrightarrow{\text{render}} \text{run}(\text{Mary}) && (\text{Mr}) \\ \text{Mary reads the book} &\xrightarrow{\text{render}} \text{read}(\text{the}(\text{book}))(\text{Mary}) && (\text{Mrb}) \end{aligned}$$

The Montague Extension of ‘Mary reads the book’ at a state $a : s$ is computed by applying the definition in the previous section⁹:

$$\begin{aligned} &\text{den}_{\text{LIL}}(\text{read}(\text{the}(\text{book}))(\text{Mary}))(a) \\ &= \text{den}_{\text{LIL}}(\text{read})(a)(\text{den}_{\text{LIL}}(\text{the})(a)(\text{den}_{\text{LIL}}(\text{book})(a)))(\text{den}_{\text{LIL}}(\text{Mary})(a)) \\ &= \text{read}(a)(\text{the}(a)(\text{book}(a)))(\text{Mary}(a)). \end{aligned}$$

The meaning of the sentence ‘Mary reads the book’ is modeled thus by a function (Montague Intension) which at every state (every context of reference) gives a truth value computed by the corresponding Montague Extensions of its constituent parts at that state.

Consider now the sentence ‘She runs’ (Sr) at a state a where $\text{Mary}(a) = \text{she}(a)$, that is, the demonstrative ‘She’ denotes the same person as the name ‘Mary’. The sentence is rendered in an analogous way and its denotation at a is shown below.

$$\begin{aligned} &\text{She runs} \xrightarrow{\text{render}} \text{run}(\text{she}) \\ &\text{den}_{\text{LIL}}(\text{run}(\text{she}))(a) = \text{run}(a)(\text{she}(a)). \end{aligned}$$

At state a , the Montague Extensions of the sentences (Sr) and (Mr) are of course equal — they are both equal to either Truth or Falsity. Their Montague Intensions are not equal functions unless $\text{Mary} : s \rightarrow e$ and $\text{she} : s \rightarrow e$ are themselves equal functions which is not of course the intended interpretation. Thus, the notion of situated meaning of a sentence at a state is exhausted by its Extension at that state.

⁹We omit assignments since the term is closed.

1.4 David Kaplan: Logic of Demonstratives

D. Kaplan's work adds considerably to the tradition of Montague semantics, but here we confine our presentation to the *Logic of Demonstratives* (LD) ([14]). Kaplan draws attention to some very interesting semantical distinctions involving utterances to motivate the need for a non trivial notion of situated meaning¹⁰. We have to point out that in general, D. Kaplan's theory of direct reference ([15]) treats *indexicality* and *demonstratives*' usage in a way that sheds light on their nature and, thus, it must be considered in any study of situated meaning.

The formal language LD “is based on first-order predicate logic with identity and descriptions”. Its semantics is based on a dichotomization of context of reference (or state) into two independent parts¹¹: the *context of utterance* and the *possible circumstance* considered as *context of evaluation*.

A context of utterance c in \mathcal{C} (the set of contexts) comprises:

- (i) an agent c_A in \mathcal{U} (the set of all individuals),
- (ii) a moment of time c_T in \mathcal{T} (the set of natural numbers, considered as moments of time),
- (iii) a position c_P in \mathcal{P} (the set of positions) and
- (iv) a world c_W in \mathcal{W} (the set of worlds).

On the other hand, a context of evaluation is a pair of

- (i) a world w in \mathcal{W} and
- (ii) a moment of time t in \mathcal{T} .

Other indicators are possible if other aspects of a circumstance are relevant to the particular situation we are referring to.

Now, in an interpretation structure of LD, \mathfrak{A} , each term A at a particular 3-tuple $\langle c, t, w \rangle$ is assigned a *denotation* belonging to an appropriate set of objects¹²

$$| A |_{\langle c, t, w \rangle}^{\mathfrak{A}} .$$

If A renders a natural language sentence, its denotation is Truth or Falsity.

¹⁰A short exposition of these ideas can also be found in [13].

¹¹There is a similar idea by Montague in [17] where the Montague Intension (modeling meanings) is defined with respect to a pair $\langle i, j \rangle$ where i is a possible word (or a pair of possible world and a moment of time) and j a context of use. It is stated that “*The second argument is introduced in order to permit a treatment,..., of such indexical locutions as demonstratives, first- and second-person singular pronouns,...*”.

¹²As usual, the denotation of each term is given with respect to an assignment to the free variables that occur in it, but we silently omit this here to focus on more important aspects of LD.

The global meaning of a term A is modeled by the *Character* of A , i.e. the function

$$\text{Character of } A : \mathcal{C} \rightarrow (\mathcal{T} \times \mathcal{W} \rightarrow \text{Denotation}(A))$$

while the *Content* of A at a particular context of utterance c is

$$\text{Content of } A \text{ at context } c : \mathcal{T} \times \mathcal{W} \rightarrow \text{Denotation}(A)(c).$$

It is thus natural that

$$\text{Character of } A = (c \mapsto \text{Content of } A \text{ at } c).$$

To refer to an example of [14], the Content of ‘I was insulted yesterday’ at a particular context c_1 is different from that of the utterance of the same sentence at another context c_2 , while of course the two sentences may have the same denotation (for example, they can be both true).

$$\begin{aligned} &\text{Content(‘I was insulted yesterday’) at } c_1 \\ &\quad \neq \text{Content(‘I was insulted yesterday’) at } c_2. \end{aligned}$$

The denotation of indexicals depends on context and as a consequence, the Character of a sentence into which an indexical occurs is not a constant function over contexts.

On the other hand, at a given context c where c_A is the individual named ‘David Kaplan’ and c_T is 21 April 1973, the Content of ‘I was insulted yesterday’ is the same as the Content of ‘David Kaplan is insulted on 20 April 1973’ at any context c' .

$$\begin{aligned} &\text{Content(‘I was insulted yesterday’) at } c \\ &\quad = \text{Content(‘David Kaplan is insulted on 20 April 1973’) at } c'. \end{aligned}$$

The two sentences have of course different Characters.

$$\begin{aligned} &\text{Character(‘I was insulted yesterday’)} \\ &\quad \neq \text{Character(‘David Kaplan is insulted on 20 April 1973’)}. \end{aligned}$$

The results of Kaplan’s logic of demonstratives will be discussed again in Section 1.6 in comparison to the two notions of situated meaning proposed there in an extension of Montague’s LIL. Finally, in Section 5.2, we will reconsider Kaplan’s ideas in the Theory of Referential Intentions using both local meaning and the proposed notion of factual content.

Names, indexicals	Mary, I, she : \tilde{e}
Common nouns	man : $\tilde{e} \rightarrow \tilde{t}$
Intransitive verbs	run, rise : $\tilde{e} \rightarrow \tilde{t}$
Transitive verbs	love : $\tilde{e} \rightarrow (\tilde{e} \rightarrow \tilde{t})$
Articles	the : $(\tilde{e} \rightarrow \tilde{t}) \rightarrow \tilde{e}$
(Basic) necessity operator	$\Box : \tilde{t} \rightarrow \tilde{t}$

Table 1.2: Constants in $L_{ar}^\lambda(K)$.

1.5 Theory of Referential Intensions: Algorithmic Formal Semantics

This section is a concise introductory presentation of the Theory of Referential Intensions which is the main technical framework of the thesis. It is presented in detail in [22] (see also [20]) and in what follows, we summarize the syntax and semantics of the formal language L_{ar}^λ in which the theory is developed.

1.5.1 Syntax and Semantics of L_{ar}^λ

$L_{ar}^\lambda(K)$ is a typed λ -calculus (Section 1.2) enriched with a recursive construct. *Types* are defined recursively by

$$\tau ::= e \mid t \mid s \mid (\tau_1 \rightarrow \tau_2) \quad (1.6)$$

where, unlike the types in LIL (1.4), the type s of *states* is a regular basic type.

The set K is a finite set of typed constants $c : \tau$. As in LIL, they introduce in $L_{ar}^\lambda(K)$ the part of the lexicon of the natural language we are interested in. In Table 1.2, some constants of K and their corresponding types are shown. The types of the constants are a subset of (1.6) defined by

$$\tilde{\sigma} ::= (s \rightarrow e) \mid (s \rightarrow t) \mid (\tilde{\sigma}_1 \rightarrow \tilde{\sigma}_2) \quad (1.7)$$

and for short, $\tilde{e} ::= (s \rightarrow e)$ and $\tilde{t} ::= (s \rightarrow t)$. Notice that, unlike the constants in LIL (Table 1.1), the constants in L_{ar}^λ are typed by using the type of states (s) in full (see also discussion in Section 2.1).

In this language, apart from the usual typed quantifiable variables (called *pure*), there are also *recursive* variables with a distinct role made clear in the recursive construct.

The *terms* of $\mathbb{L}_{\text{ar}}^\lambda(K)$ are defined by

$$A ::= c \mid x \mid B(C) \mid \lambda(v)(B) \mid A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\} \quad (1.8)$$

where x is a pure or recursive variable, v is a pure variable and p_1, \dots, p_n are recursive variables. A type is assigned to each term by this definition and *free* and *bound* occurrences of the variables are determined.

In what concerns the recursive term, we note that

- (i) p_1, \dots, p_n are distinct recursive variables and they occur bound in it,
- (ii) for all $i = 1, \dots, n$, $A_i, p_i : \tau_i$ and if $A_0 : \tau$, then

$$A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\} : \tau,$$

(iii) and, the system $\{p_1 := A_1, \dots, p_n := A_n\}$ is *acyclic* — that is, we can associate a natural number, $\text{rank}(p_i)$, with each recursive variable p_i so that if p_j occurs free in A_i , then $\text{rank}(p_i) > \text{rank}(p_j)$.

In $\mathbb{L}_{\text{ar}}^\lambda$, we define *congruence* as the smallest equivalence relation \equiv_c between terms which respects alphabetic replacement of bound, pure and recursive, variables, application, λ -abstraction and acyclic recursion and such that for any permutation $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$, for any recursive term

$$\begin{aligned} A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\} \\ \equiv_c A_0 \text{ where } \{p_{\pi(1)} := A_{\pi(1)}, \dots, p_{\pi(n)} := A_{\pi(n)}\}. \end{aligned}$$

Both LIL and $\mathbb{L}_{\text{ar}}^\lambda$ are formal languages at the core of which lies the typed λ -calculus (Section 1.2). Apart from the differences in types, in LIL there are two additional operators, the intension and the extension ones, while in $\mathbb{L}_{\text{ar}}^\lambda$ there is a new recursive construct. Without trying to compare them, there is a simple point worth mentioning¹³. The intensional operator enriches the λ -calculus in a basic way: with its use, one can define formally within LIL the semantic value (Montague Intension) that is expressed by its terms. We will see in what follows that the recursive construct plays an analogous role in $\mathbb{L}_{\text{ar}}^\lambda$. This is an important feature of both languages and reveals rather than obscures their similarity.

The following examples are terms in $\mathbb{L}_{\text{ar}}^\lambda$ which render natural language sentences in exactly the same way as in LIL. Notice though that these terms are now of type $\tilde{\mathbf{t}}$, and not of type \mathbf{t} .

$$\text{Mary runs} \xrightarrow{\text{render}} \text{run}(\text{Mary}) \quad (\text{Mr})$$

$$\text{Mary reads the book} \xrightarrow{\text{render}} \text{read}(\text{the}(\text{book}))(\text{Mary}) \quad (\text{Mrb})$$

¹³See also the discussion at the end of this section.

Finally, the following are examples of two recursive terms in $\mathbb{L}_{\text{ar}}^\lambda$ again of type $\tilde{\mathbf{t}}$.

$$\begin{aligned} & \text{run}(p) \text{ where } \{p := \text{Mary}\} \\ & \text{read}(p_1)(p_3) \text{ where } \{p_1 := \text{the}(p_2), p_2 := \text{book}, p_3 := \text{Mary}\} \end{aligned}$$

An interpretation structure \mathcal{U} of $\mathbb{L}_{\text{ar}}^\lambda(K)$ comprises:

- (i) non empty universes of objects of basic and function types, $\mathbb{T}_{\mathbf{e}}$, $\mathbb{T}_{\mathbf{t}}$ and $\mathbb{T}_{\mathbf{s}}$ and $\mathbb{T}_{\tau_1 \rightarrow \tau_2} = \{f \mid f : \mathbb{T}_{\tau_1} \rightarrow \mathbb{T}_{\tau_2}\}$,
- (ii) an object $c \in \mathbb{T}_\tau$ for every constant $\mathbf{c} : \tau$ of K .

It is useful to assume that there are special objects, $er_{\mathbf{e}} \in \mathbb{T}_{\mathbf{e}}$ and $er_{\mathbf{t}} \in \mathbb{T}_{\mathbf{t}}$ which represent the value “error” for entities and truth values, respectively. They build up for all types, that is, there is an object er_σ for every type σ and thus, we can compute the value of the denotation for every term.

In \mathcal{U} , we define a denotation function den that associates with each term $A : \tau$ and each assignment g to the variables, the object $\text{den}(A)(g) \in \mathbb{T}_\tau$ by the recursion:

- $\text{den}(x)(g) = g(x); \text{den}(\mathbf{c})(g) = c.$
- $\text{den}(A(B))(g) = \text{den}(A)(g)(\text{den}(B)(g)).$
- $\text{den}(\lambda(v)(B))(g) = (f \mapsto \text{den}(B)(g\{v := f\})),$
where if $v : \tau$, then f is any object in \mathbb{T}_τ .
- $\text{den}(A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\})(g)$
 $= \text{den}(A_0)(g\{p_1 := P_1, \dots, p_n := P_n\})$
where for $i = 1, \dots, n$, if p_{j_1}, \dots, p_{j_m} are the recursive variables with rank lower than $\text{rank}(p_i)$, each object P_i is defined by

$$P_i = \text{den}(A_i)(g\{p_{j_1} := P_{j_1}, \dots, p_{j_m} := P_{j_m}\}).$$

Apart from the denotational semantics, we will associate with each *proper* term $A : \sigma$ its *referential intension* $\text{int}(A)$ which models its global meaning in $\mathbb{L}_{\text{ar}}^\lambda(K)$,

$$A \mapsto \text{int}(A).$$

In what follows, we describe briefly how this association is established.

First, we call *proper* the terms that are not *immediate*. Immediate terms in $\mathbb{L}_{\text{ar}}^\lambda$ are treated like generalized variables and they are not assigned meanings. They are defined by

$$X \equiv v_i \mid p(v_1, \dots, v_n) \mid \lambda(u_1, \dots, u_m)p(v_1, \dots, v_n) \quad (1.9)$$

where p is a recursive variable and v_1, \dots, v_n and u_1, \dots, u_m are pure variables. Notice that by this definition for any recursive variable p , both p itself (considered as $p = \lambda(u_1, \dots, u_m)p(u_1, \dots, u_m)$) and any λ -term of the form $\lambda(u_1, \dots, u_m)p$ are immediate terms.

Second, in L_{ar}^λ , a *reduction* relation $A \Rightarrow B$ on terms is defined by ten *reduction rules*, presented in Table 1.3. Reduction rules are considered as “compilation” rules in this theory and it is proved in [22] that they respect denotational equivalence, their application terminates always and its result is unique up to congruence.

We also define *irreducible* terms with respect to this reduction relation.

$$A \text{ is irreducible} \iff \text{for all } B, \text{ if } A \Rightarrow B, \text{ then } A \equiv_c B. \quad (1.10)$$

Theorem 1.5.1. (Canonical Form Theorem) *For each term A , there is a unique (up to congruence) irreducible recursive term*

$$\text{cf}(A) \equiv A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\}$$

such that $A \Rightarrow \text{cf}(A)$.

The *canonical form* $\text{cf}(A)$ can be effectively computed from A , it is denotationally equivalent with A , and its *head* A_0 and its *parts* A_1, \dots, A_n determine the basic computable modules that are needed in order to compute the denotation of the original term A .

Now, the *referential intension* of A is the tuple of functions

$$\text{int}(A) = (f_0, f_1, \dots, f_n)$$

defined by the head and the parts of its canonical form, where

$$f_i(d_1, \dots, d_n, g) = \text{den}(A_i)(g\{p_1 := d_1, \dots, p_n := d_n\}) \quad (i = 0, \dots, n).$$

Tuples of functions like the one above are called *recursors*. They are defined and used as formal representations of *abstract algorithms* in a series of papers by Y. N. Moschovakis, see [21] and earlier articles referenced there¹⁴.

For any term A , its referential intension, $\text{int}(A)$, models the (*global*) *meaning* of A in this theory — the abstract algorithm that computes its denotation, $\text{den}(A)(g)$. Thus, *the canonical form of A defines formally within the language the meaning of A .*

Global Meaning of A : $\text{cf}(A)$.

¹⁴A short, but highly motivating, introduction of these ideas can also be found in the last section of [22].

-
- (**cong**) If $A \equiv_c B$, then $A \Rightarrow B$
- (**trans**) If $A \Rightarrow B$ and $B \Rightarrow C$, then $A \Rightarrow C$
- (**rep1**) If $A \Rightarrow A'$ and $B \Rightarrow B'$, then $A(B) \Rightarrow A'(B')$
- (**rep2**) If $A \Rightarrow B$, then $\lambda(u)(A) \Rightarrow \lambda(u)(B)$
- (**rep3**) If $A_i \Rightarrow B_i$ for $i = 0, \dots, n$, then
 A_0 where $\{p_1 := A_1, \dots, p_n := A_n\}$
 $\Rightarrow B_0$ where $\{p_1 := B_1, \dots, p_n := B_n\}$
- (**head**) $(A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\})$ where
 $\{q_1 := B_1, \dots, q_m := B_m\}$
 $\Rightarrow A_0$ where $\{p_1 := A_1, \dots, p_n := A_n, q_1 := B_1, \dots, q_m := B_m\}$
- (**B-S**) A_0 where $\{p := (B_0 \text{ where } \{q_1 := B_1, \dots, q_m := B_m\}),$
 $p_1 := A_1, \dots, p_n := A_n\}$
 $\Rightarrow A_0$ where $\{p := B_0, q_1 := B_1, \dots, q_m := B_m,$
 $p_1 := A_1, \dots, p_n := A_n\}$
- (**recap**) $(A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\})(B)$
 $\Rightarrow A_0(B)$ where $\{p_1 := A_1, \dots, p_n := A_n\}$
- (**ap**) $A(B) \Rightarrow A(b)$ where $\{b := B\}$, if B is proper
- (**λ -rule**) $\lambda(u)(A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\})$
 $\Rightarrow \lambda(u)A'_0$ where $\{p'_1 := \lambda(u)A'_1, \dots, p'_n := \lambda(u)A'_n\}$

where $(i = 1, \dots, n)$, p'_i is a fresh recursive variable and
 $A'_i := A_i\{p_1 \equiv p'_1(u), \dots, p_n \equiv p'_n(u)\}.$

Table 1.3: The Reduction Calculus.

For example, the global meaning of ‘Mary reads the book’ (Mrb) in the Theory of Referential Intentions is defined by its canonical form¹⁵

$$\begin{aligned} & \text{read}(\text{the}(\text{book}))(\text{Mary}) \\ & \Rightarrow_{\text{cf}} \text{read}(p_1)(p_3) \text{ where } \{p_1 := \text{the}(p_2), p_2 := \text{book}, p_3 := \text{Mary}\}. \end{aligned}$$

It determines the particular step-by-step computation that we need to follow to compute the denotation of (Mrb). Notice that the type of the term $\text{read}(\text{the}(\text{book}))(\text{Mary})$ is $\tilde{\mathbf{t}}$ and thus the denotation that we compute through its canonical form gives its truth value at every possible state. The meaning is characterized as global exactly because it expresses the semantical value that one knows if she knows the language.

A final note about the comparison with respect to semantics, between $\mathbf{L}_{\text{ar}}^\lambda$ and the typed λ -calculus. First of all, $\mathbf{L}_{\text{ar}}^\lambda$ is not denotationally more expressive than the typed λ -calculus. It can be proved that every term is denotationally equal to an *explicit* term, that is, a term that has no *where* in it. On the other hand, there are terms in $\mathbf{L}_{\text{ar}}^\lambda$ that are not *referentially synonymous* with any explicit term. Referential synonymy is defined in the next section but basically this result shows that terms in $\mathbf{L}_{\text{ar}}^\lambda$ express more “meanings” than λ -calculus terms (see also paragraphs 1.7 and 3.24-3.25 in [22] for relevant discussion).

1.5.2 Referential Synonymy and Local Meaning

In the Theory of Referential Intentions, two terms are referentially synonymous if their referential intentions are naturally isomorphic. Nevertheless, we don’t have to use the definition of *natural isomorphism* between recursors, but instead we will use the following (equivalent) definition based on canonical forms.

Definition 1.5.2. For any two terms A and B , A is *referentially synonymous with* B ($A \approx B$) if and only if

$$\begin{aligned} & A \Rightarrow_{\text{cf}} A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\} \\ & B \Rightarrow_{\text{cf}} B_0 \text{ where } \{p_1 := B_1, \dots, p_n := B_n\} \end{aligned}$$

and for each $i = 0, \dots, n$ and all g , $\text{den}(A_i)(g) = \text{den}(B_i)(g)$.

Thus, we can decide on the synonymy between two terms based on the denotational equivalence of the corresponding parts and head of their canonical forms.

¹⁵We simply note $A \Rightarrow_{\text{cf}} B$ if and only if $B \equiv_c \text{cf}(A)$.

Referential synonymy is a structural equivalence relation which can make semantical distinctions between denotationally equal expressions. For example, with (Mr) and (Sr) as above, since $Mary \neq she$,

$$\text{run}(q) \text{ where } \{q := \text{Mary}\} \not\approx \text{run}(q) \text{ where } \{q := \text{she}\},$$

and hence, $\text{run}(\text{Mary}) \not\approx \text{run}(\text{she})$.

Consider also the sentence (Mrb) and the sentence ‘Mary reads the small book’. It is straightforward that

$$\begin{aligned} & \text{read}(p_1)(p_3) \text{ where } \{p_1 := \text{the}(p_2), p_2 := \text{book}, p_3 := \text{Mary}\} \\ & \not\approx \text{read}(p_1)(p_3) \text{ where } \{p_1 := \text{the}(p_2), p_2 := \text{small}(p_4), p_4 := \text{book}, p_3 := \text{Mary}\} \end{aligned}$$

since the two canonical forms do not have the same number of parts. The inequality of the number of parts reveals an important difference in computation which referential synonymy takes into account.

An analogous synonymy relation in LIL would express equality between the corresponding Montague Intensions of the sentences. Structural characteristics of the sentences as those described above do not affect this kind of synonymy. This is a feature of the Theory of Referential Intensions that makes a definite turn on the way truth-conditional theories of meaning are considered and developed.

Consider now the sentence (Mr) uttered at a particular context of reference, at state $a : s$. Its denotation,

$$\text{den}(\text{run}(\text{Mary}))(a) = \text{run}(\text{Mary})(a),$$

is Truth or Falsity, depending on whether the entity $\text{Mary}(a)$ is actually running at state a . In the Theory of Referential Intensions, for terms of type \tilde{t} , besides denotations, we can also define a non trivial *local meaning*. To do that, it is convenient to enrich the language with a parameter \bar{a} associated with each state a such that for any assignment g , $\text{den}(\bar{a})(g) = g(\bar{a}) = a$. Now, we define

$$\text{Local Meaning of } A : \tilde{t} \text{ at state } a \equiv \text{int}(A(\bar{a}))$$

and the corresponding relation of *local synonymy*:

Definition 1.5.3. For any two terms A and B , A at state a is *locally synonymous* with B at state b if and only if

$$\begin{aligned} A(\bar{a}) & \Rightarrow_{\text{cf}} A_0(\bar{a}) \text{ where } \{p_1 := A_1, \dots, p_n := A_n\} \\ B(\bar{b}) & \Rightarrow_{\text{cf}} B_0(\bar{b}) \text{ where } \{p_1 := B_1, \dots, p_n := B_n\} \end{aligned}$$

and for all g , $\text{den}(A_0)(g)(\bar{a}) = \text{den}(B_0)(g)(\bar{b})$ and for each $i = 1, \dots, n$, $\text{den}(A_i)(g) = \text{den}(B_i)(g)$.

The local meaning of the sentence ‘Mary sleeps’ at state a is expressed by

$$\text{sleep}(q)(\bar{a}) \text{ where } \{q := \text{Mary}\}.$$

By the definition, it is locally synonymous with the utterance of ‘Mary relaxes’ at the same state a , i.e.,

$$\text{sleep}(q)(\bar{a}) \text{ where } \{q := \text{Mary}\} \approx \text{relax}(q)(\bar{b}) \text{ where } \{q := \text{Mary}\},$$

if and only if for any assignment g ,

$$\text{sleep}(g(p))(a) = \text{relax}(g(p))(a),$$

that is, if the set of individuals that sleep at state a is the same as the set of the individuals that relax at that same state.

Suppose now that at state a , $\text{Mary}(a) = \text{she}(a)$, and consider again the utterances of the two sentences (Mr) and (Sr). Can we deduce that the two utterances are locally synonymous? The answer is negative,

$$\text{run}(q)(\bar{a}) \text{ where } \{q := \text{Mary}\} \not\approx \text{run}(q)(\bar{a}) \text{ where } \{q := \text{she}\},$$

since $\text{Mary} \neq \text{she}$. Notice that the local referential intensions of these utterances depend on the global meaning of **Mary** and **she**. In general, local synonymy between two utterances at some state is a structural notion of synonymy that is not exhausted by the denotational equivalence of the corresponding parts at that state.

The need of an additional notion of meaning which takes into account the state dependent denotational equalities of the parts of the canonical form of a term leads to the study of *factual content* which is the subject of this thesis (see Introduction and Chapter 5 for relevant discussion). It is conceived here not as an amendment to the local meaning but as its counterpart towards a complete framework of situated meaning.

1.6 Two Notions of Situated Meaning

In this last section of the chapter, we will explore the possibility to enrich Ty_2 , a two-sorted typed λ -calculus presented in [8], with a recursive construct, define referential intensions in it and then, exploiting the way the terms of

this language are typed, define, in addition to global and local meaning, a non trivial notion of factual content¹⁶.

Ty_2 is formed by translating each term A of LIL and each state variable u to a term $A^{G,u}$ of the same type as in LIL, where, though, the state variable u is made explicit,

$$A : \tau \text{ and a state variable } u \mapsto A^{G,u} : \tau.$$

In what follows, we won't focus on the translation but rather present directly this new language and compare it to LIL and $\mathcal{L}_{\text{ar}}^\lambda$.

Types in Ty_2 are the types of $\mathcal{L}_{\text{ar}}^\lambda$ (1.6), that is, unlike LIL, states form a basic type of their own. As in Section 1.2, *terms* are defined by

$$A ::= x \mid \mathbf{c}^G \mid A(B) \mid \lambda(x)(B). \quad (1.11)$$

Semantically, we interpret Ty_2 terms (which form a subset of $\mathcal{L}_{\text{ar}}^\lambda$ terms) in an interpretation structure similar to that of $\mathcal{L}_{\text{ar}}^\lambda$, with the denotation function defined as usual. Thus, unlike LIL, the terms in Ty_2 are such that for each term $A : \tau$ and each assignment g to the variables, $\text{den}(A)(g) \in \mathbb{T}_\tau$.

Let us now consider again the constants of LIL that we introduced in Table 1.1. We presented there only constants that are extensional in order to focus on the rendering procedure but these are just a subset of natural language constants. Montague actually believed that intensional phenomena are the rule and that is why he defined functional application by means of the Montague intension of the argument. We don't follow this idea in LIL but we rather type differently the constants according to the way they treat their arguments¹⁷. For example, notice the type of *rise* in Table 1.4.

Now, for each constant $\mathbf{c} : \tau$ in LIL, $\mathbf{c}^G : \mathbf{s} \rightarrow \tau$ is introduced in Ty_2 such that $\text{den}(\mathbf{c}^G) = c$. The Gallin translation of any constant $\mathbf{c} : \tau$ at a state u in Ty_2 is $\mathbf{c}^G(u) : \tau$, so that for any assignment g such that $g(u) = a$,

$$\text{den}(\mathbf{c}^{G,u})(g\{u := a\}) = \text{den}(\mathbf{c}^G(u))(g\{u := a\}) = c(a) = \text{den}_{\text{LIL}}(\mathbf{c})(g)(a).$$

In general, for each LIL term $A : \tau$, $A^{G,u} : \tau$, and for any assignment g such that $g(u) = a$,

$$\text{den}(A^{G,u})(g) = \text{den}_{\text{LIL}}(A)(g)(a).$$

The idea is that for each A and each state a , *its Gallin translation* $A^{G,u}$ *defines formally within* Ty_2 *its LIL denotation at* a .

¹⁶This work is presented in detail in [13]. A short account of it along with some preliminary results of this thesis can also be found in [12].

¹⁷It is no longer true that constants of the same syntactical category are typed uniformly.

Extensional intransitive verbs	$\text{run} : e \rightarrow t$
Intensional intransitive verbs	$\text{rise} : (s \rightarrow e) \rightarrow t$
Extensional transitive verbs	$\text{love} : e \rightarrow (e \rightarrow t)$
(Basic) necessity operator	$\Box : (s \rightarrow t) \rightarrow t$
de dicto modal operator	$\text{Yesterday} : (s \rightarrow t) \rightarrow t$
de re modal operator	$\text{Yesterday}_1 : (s \rightarrow (e \rightarrow t)) \rightarrow (e \rightarrow t)$

Table 1.4: More constants in LIL.

We now form $L_{ar}^{\lambda, G}$ by adding to Ty_2 an infinite number of recursive variables for each type τ , and extending the definition of terms by adding the recursive construct as follows

$$A ::= x \mid p \mid c^G \mid A(B) \mid \lambda(x)(B) \mid A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\}. \quad (1.12)$$

Notice that $L_{ar}^{\lambda, G}$ is noted in [13] as simply L_{ar}^{λ} because the original L_{ar}^{λ} (Section 1.5) is not presented there. We introduce this new notation in order to avoid confusion although it should be clear that, if we exclude the typing of the constants, the two languages are basically the same.

It is now possible to define in $L_{ar}^{\lambda, G}$ for any LIL term A , aside from its global and local meaning, an additional situated meaning, its *factual content*. We have already pointed out that for any LIL term A , its Gallin translation $A^{G, u}$ with respect to a state variable u defines formally the denotation of A at a state. Now, the abstraction over all states of the Gallin translation of A defines formally the denotation of A and, thus, the canonical form of that term defines the algorithm which computes the denotation of A . That is,

$$\text{Global Meaning of } A : \quad \text{cf}(\lambda(u)(A^{G, u}))$$

As in Section 1.5.2, by evaluation, we simply have for any LIL term A and a state a

$$\text{Local Meaning of } A \text{ at state } a : \quad \text{cf}(\lambda(u)(A^{G, u})(\bar{a})).$$

Notice that this definition applies to LIL terms of any type and not just for terms of type \tilde{t} as stated in Section 1.5.2.

Since β -conversion does not preserve referential synonymy, in general

$$\lambda(u)(A^{G, u})(\bar{a}) \not\approx A^{G, u}\{u := \bar{a}\},$$

the canonical form of $A^{G,u}\{u := \bar{a}\} \equiv A^{G,\bar{a}}$ defines an alternative notion of situated meaning of A in $\mathbb{L}_{\text{ar}}^{\lambda,G}$,

$$\text{Factual Content of } A \text{ at state } a : \text{cf}(A^{G,\bar{a}}).$$

For example, in the case of the simple sentence (Mr), considered as a LIL term, the three notions of meaning are defined by the following canonical forms:

$$\begin{aligned} \text{Global Meaning} & : \lambda(u)(\text{run}^G(u)(q(u))) \text{ where } \{q := \lambda(u)\text{Mary}^G(u)\} \\ \text{Local Meaning at } a & : \lambda(u)(\text{run}^G(u)(q(u))) (\bar{a}) \text{ where } \{q := \lambda(u)\text{Mary}^G(u)\} \\ \text{Factual Content at } a & : \text{run}^G(\bar{a})(q') \text{ where } \{q' := \text{Mary}^G(\bar{a})\} \end{aligned}$$

If at a state a , $\text{she}^G(a) = \text{Mary}^G(a)$, this notion of factual content produces the expected synonymy between the utterances of (Mr) and (Sr).

$$\text{run}^G(\bar{a})(q') \text{ where } \{q' := \text{Mary}^G(\bar{a})\} \approx \text{run}^G(\bar{a})(q') \text{ where } \{q' := \text{she}^G(\bar{a})\}.$$

It is again true that the two utterances are not locally synonymous

$$\begin{aligned} \lambda(u)(\text{run}^G(u)(q(u))) (\bar{a}) \text{ where } \{q := \lambda(u)\text{Mary}^G(u)\} \\ \approx \lambda(u)(\text{run}^G(u)(q(u))) (\bar{a}) \text{ where } \{q := \lambda(u)\text{she}^G(u)\} \end{aligned}$$

which agrees with our intuition that these sentences *convey the same information about the world* at that particular context of reference but *the computation of that information involves different basic computational modules*.

Let us now consider the more interesting Kaplan example ‘I was insulted yesterday’. In LIL, it is rendered as¹⁸

$$\text{I was insulted yesterday} \xrightarrow{\text{render}} \text{Yesterday}_1 (\wedge (\text{be_insulted})) (I)$$

where $\text{be_insulted} : e \rightarrow t$. Its proposed factual content at any state a is defined by

$$\text{Yesterday}_1^G(\bar{a})(p)(q) \text{ where } \{p := \lambda(u)\text{be_insulted}^G(u), q := I^G(\bar{a})\}$$

and, if at state b , it is uttered by a different person ($I^G(a) \neq I^G(b)$), then

$$\begin{aligned} \text{Yesterday}_1^G(\bar{a})(p)(q) \text{ where } \{p := \lambda(u)\text{be_insulted}^G(u), q := I^G(\bar{a})\} \\ \approx \text{Yesterday}_1^G(\bar{b})(p)(q) \text{ where } \{p := \lambda(u)\text{be_insulted}^G(u), q := I^G(\bar{b})\}. \end{aligned}$$

¹⁸The word ‘Yesterday’ is rendered here as the constant Yesterday_1 and not as Yesterday since the existence of the indexical ‘I’ forces the use of the de re version of it.

Suppose that $\text{DavidKaplan} : e$ is a constant in LIL that denotes David Kaplan and $\text{on20April1973} : ((s \rightarrow (e \rightarrow t)) \rightarrow (e \rightarrow t))$ is an intensional operator that determines time. It is trivially the case that if the time at state a is 21 April 1973 and the speaker is David Kaplan, then at any state b , where $\text{DavidKaplan}^G(\bar{b})$ denotes David Kaplan,

$$\begin{aligned} & \text{Yesterday}_1^G(\bar{a})(p)(q) \text{ where } \{p := \lambda(u)\text{be_insulted}^G(u), q := \text{!}^G(\bar{a})\} \\ & \approx \text{on20April1973}^G(\bar{b})(p)(q) \text{ where} \\ & \quad \{p := \lambda(u)\text{be_insulted}^G(u), q := \text{DavidKaplan}^G(\bar{b})\}. \end{aligned}$$

Apart from Kaplan and his ideas on situated meaning, in [13], there is a more general discussion on the interrelations of the three notions of meaning defined for terms in LIL. We will return on some of these considerations on Chapter 5 for terms of L_{ar}^λ and with the aid of the proposed notion of factual content that will be defined there.

Chapter 2

Locality of Typed Objects

In this chapter, we introduce the basic notions about *locality* for the objects in the interpretation structure of $\mathsf{L}_{\text{ar}}^\lambda$ (cf. Section 1.5.1). First, we identify the objects that interpret terms that render natural language expressions and thus, they are the objects whose locality behavior we are interested in. In $\mathsf{L}_{\text{ar}}^\lambda$, these objects are typed by the types in (1.7) and this is made precise in Section 2.1.

In Section 2.2, we define *local objects* — their values at a state depend only on the values of their arguments at that state. They are characterized by *local associates* whose properties are presented.

In Section 2.3, the notions of the previous section are generalized for all objects of natural language renderings and locality is deployed in its full complexity. *Locality indices* — strings that codify the locality behavior of objects — are introduced and with the use of them, the notion of an *associate* of an object.

2.1 State-dependent Types of $\mathsf{L}_{\text{ar}}^\lambda$

Types of $\mathsf{L}_{\text{ar}}^\lambda$ were defined by (1.6) in Chapter 1 by the recursion

$$\tau ::= \mathbf{e} \mid \mathbf{t} \mid \mathbf{s} \mid (\tau_1 \rightarrow \tau_2). \quad (2.1)$$

In order to define locality, we will focus on some interesting subsets of these types. First of all, the *pure* or *state-free* types are the $\mathsf{L}_{\text{ar}}^\lambda$ types without the basic type \mathbf{s} ,

$$\sigma ::= \mathbf{e} \mid \mathbf{t} \mid (\sigma_1 \rightarrow \sigma_2). \quad (2.2)$$

It is not uncommon to use these types in natural language rendering since it is often assumed that any interpretation is performed at a particular fixed

state, “the current state” (see also footnote 8 in page 12). Montague types follow this idea and the type \mathbf{s} appears only in the types $(\mathbf{s} \rightarrow \tau)$ which are used to define the Montague Intension of a term in LIL (cf. (1.4) in Section 1.3).

On the other hand, the *state-dependent* types of $\mathbf{L}_{\text{ar}}^\lambda$, defined in (1.7), are types all of whose parts depend on states,

$$\tilde{\sigma} \equiv (\mathbf{s} \rightarrow \mathbf{e}) \mid (\mathbf{s} \rightarrow \mathbf{t}) \mid (\tilde{\sigma}_1 \rightarrow \tilde{\sigma}_2). \quad (2.3)$$

These two subset of $\mathbf{L}_{\text{ar}}^\lambda$ types are more closely related than it appears at first sight. The following simple transformation adds states uniformly in the pure types and its converse removes states from the state-dependent ones. For each pure type σ , we associate the state-dependent type $\tilde{\sigma}$ recursively by

$$\begin{aligned} \tilde{\mathbf{e}} &\equiv (\mathbf{s} \rightarrow \mathbf{e}) \\ \tilde{\mathbf{t}} &\equiv (\mathbf{s} \rightarrow \mathbf{t}), \text{ and} \\ \text{if } \sigma &\equiv (\sigma_1 \rightarrow \sigma_2), \text{ then } \tilde{\sigma} \equiv (\tilde{\sigma}_1 \rightarrow \tilde{\sigma}_2). \end{aligned}$$

For example, if $\tilde{\sigma} \equiv (\tilde{\mathbf{e}} \rightarrow \tilde{\mathbf{t}}) \rightarrow (\tilde{\mathbf{e}} \rightarrow \tilde{\mathbf{t}})$, then the corresponding pure type σ is $(\mathbf{e} \rightarrow \mathbf{t}) \rightarrow (\mathbf{e} \rightarrow \mathbf{t})$. And if for example $\sigma \equiv (\mathbf{t} \rightarrow \mathbf{t}) \rightarrow (\mathbf{e} \rightarrow (\mathbf{e} \rightarrow \mathbf{t}))$, the corresponding state-dependent $\tilde{\sigma}$ is $(\tilde{\mathbf{t}} \rightarrow \tilde{\mathbf{t}}) \rightarrow (\tilde{\mathbf{e}} \rightarrow (\tilde{\mathbf{e}} \rightarrow \tilde{\mathbf{t}}))$.

The constants of $\mathbf{L}_{\text{ar}}^\lambda(K)$ in Table 1.2 have state-dependent types. Terms that render natural language expressions are built up from these constants and thus they are typed by exactly the types in (2.3). It will be clear in the following sections of this chapter that our study will be confined on functions of state-dependent type (and on Chapter 3 on analogously typed terms).

Finally, as usual, we define for any type in (2.1)

$$\tau_1 \times \tau_2 \rightarrow \tau_3 \equiv \tau_1 \rightarrow (\tau_2 \rightarrow \tau_3)$$

which we can generalize for any number of types¹. Based on that, we can alternatively define the general form of a type τ of $\mathbf{L}_{\text{ar}}^\lambda$ as

$$\tau \equiv \tau_1 \times \dots \times \tau_n \rightarrow \tau_0$$

¹This is the Schönfinkelization or Currying method of how a n -place function can be reduced to a function whose arguments are 1-place functions. It is used in natural language rendering (see [10]) in order to express more directly syntax. In this approach we will exploit the equivalence of the two views and we will typically view the objects in the standard $\mathbf{L}_{\text{ar}}^\lambda$ structure as functions of many variables whose values are objects of the basic types.

where τ_1, \dots, τ_n is any type and $\tau_0 := \mathbf{e} \mid \mathbf{t} \mid \mathbf{s}$. We also associate with each type τ a natural number, the *level* of τ ($\text{level}(\tau)$), as follows

$$\begin{aligned} \text{level}(\mathbf{e}) &= \text{level}(\mathbf{t}) = \text{level}(\mathbf{s}) = 0 \\ \text{level}(\tau_1 \times \dots \times \tau_n \rightarrow \tau_0) &= \max(\text{level}(\tau_1), \dots, \text{level}(\tau_n)) + 1. \end{aligned}$$

In the case of state-dependent types, their general form is

$$\tilde{\sigma} \equiv \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0 \quad \text{where } \tilde{\sigma}_0 := \tilde{\mathbf{e}} \mid \tilde{\mathbf{t}} \quad (2.4)$$

and $\tilde{\sigma}_1, \dots, \tilde{\sigma}_n$ are state-dependent types. The basic state-dependent types are of level 1 ($\text{level}(\tilde{\mathbf{e}}) = \text{level}(\tilde{\mathbf{t}}) = 1$) whereas for any type $\tilde{\sigma}_1 \rightarrow \tilde{\sigma}_2$, $\text{level}(\tilde{\sigma}_1 \rightarrow \tilde{\sigma}_2) \geq 2$.

2.2 Local Objects

A function of a state-dependent type $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$ can be characterized by whether its value $f(f_1, \dots, f_n, a)$ at a state $a : \mathbf{s}$ depends on the values of its arguments $f_i : \tilde{\sigma}_i$ ($i = 1, \dots, n$) on states different than a or not. We will show that this is a concrete notion of **locality** and we will, first, consider in this section the case where the value of f at a does not depend on the values of its arguments at other states than a .

Example 2.2.1. Suppose we consider the object $\text{love} : \tilde{\mathbf{e}} \times \tilde{\mathbf{e}} \rightarrow \tilde{\mathbf{t}}$ which interprets the constant *love* in L_{ar}^λ . For any two objects $f_1, f_2 : \tilde{\mathbf{e}}$ and any state $a : \mathbf{s}$, the function *love* is defined by

$$\text{love}(f_1, f_2, a) = 1 \iff f_2(a) \text{ loves } f_1(a) \text{ at state } a.$$

Naturally enough, the evaluation of $\text{love}(f_1, f_2, a)$ only needs the values of f_1 and f_2 at state a . In other words, for any two pairs of arguments $(f_1, f_2 : \tilde{\mathbf{e}})$ and $(f'_1, f'_2 : \tilde{\mathbf{e}})$ and any state a , if $f_1(a) = f'_1(a)$ and $f_2(a) = f'_2(a)$, then $\text{love}(f_1, f_2, a) = \text{love}(f'_1, f'_2, a)$.

The object *love* is just one example of a *local* object and we will make this characterization formal with a use of *local associates*.

Definition 2.2.2. (Local Associate of an Object) For each $f : \tilde{\sigma}$, we define what it means for

$$f' : \mathbf{s} \rightarrow \sigma$$

to be a *local associate* of f by induction on the level of $\tilde{\sigma}$.

- (i) If $f : \tilde{\sigma}_0$, then f' is a local associate of f if and only if $f' = f$.
- (ii) If $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$, then $f' : \mathbf{s} \times \sigma_1 \times \dots \times \sigma_n \rightarrow \sigma_0$ is a local associate of f if and only if for any $f_i : \tilde{\sigma}_i$, $a : \mathbf{s}$,

$$f(f_1, \dots, f_n, a) = f'(a, f'_1(a), \dots, f'_n(a))$$

where $f'_i : \mathbf{s} \rightarrow \sigma_i$, are such that for $i = 1, \dots, n$, f'_i is a local associate of f_i .

An object $f : \tilde{\sigma}$ is *local* if it has a local associate.

The definition of *love* presented above expressed nothing more than the existence of a local associate for it and thus, the object *love* is local. It is also straightforward by the definition that all objects of types $\tilde{\mathbf{e}}$ and $\tilde{\mathbf{t}}$ (objects of level one types) are local and that they have unique local associates.

Example 2.2.3. Other trivial local objects of arbitrary level are constant functions, that is, any function $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$ such that for any f_1, \dots, f_n of appropriate types and any $a : \mathbf{s}$,

$$f(f_1, \dots, f_n, a) = b$$

for some $b : \sigma_0$. The function $f' : \mathbf{s} \times \sigma_1 \times \dots \times \sigma_n \rightarrow \sigma_0$ such that for any g_1, \dots, g_n of appropriate types and any $a : \mathbf{s}$,

$$f'(a, g_1, \dots, g_n) = b$$

is obviously a local associate of f .

Notice that if f' is an associate of both f_1 and f_2 , then f_1 and f_2 are equal on all local arguments. On the other hand, this definition does not determine $f' : \mathbf{s} \rightarrow \sigma$ — it merely determines some of its values. The definition does not impose any constraint on the values of f' on arguments that are not themselves values of local associates of arguments of f . In general, a local object $f : \tilde{\sigma}$ with $\text{level}(\tilde{\sigma}) \geq 2$ has many local associates whose relation is described in the following straightforward proposition.

Proposition 2.2.4. *If f_1 and f_2 are local associates of $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$, then, for any $a : \mathbf{s}$ and any local objects $g_i : \tilde{\sigma}_i$ and local associates $g'_i : \mathbf{s} \rightarrow \sigma_i$ of them, ($i = 1, \dots, n$),*

$$f_1(a, g'_1(a), \dots, g'_n(a)) = f_2(a, g'_1(a), \dots, g'_n(a)).$$

Let us now consider an example of a *non local* object.

Example 2.2.5. The most obvious is the sentential operator $\Box : \tilde{\mathbf{t}} \rightarrow \tilde{\mathbf{t}}$ interpreted as “necessarily always”, that is for any $f : \tilde{\mathbf{t}}$ and any state $a : \mathbf{s}$,

$$\Box(f, a) = 1 \iff \forall b : \mathbf{s}, f(b) = 1.$$

To arrive at a contradiction, assume that there is a function $f' : \mathbf{s} \times \mathbf{t} \rightarrow \mathbf{t}$ such that for any $f : \tilde{\mathbf{t}}$ and any $a : \mathbf{s}$, $\Box(f, a) = f'(f(a), a)$. Suppose also that for two objects f_1 and f_2 at state a , $f_1(a) = f_2(a) = 1$ but $\Box(f_1, a) = 1$, $\Box(f_2, a) = 0$ because there is another state $b : \mathbf{s}$ such that $f_2(b) = 0$.

Now, while $f'(f_1(a), a) = f'(f_2(a), a)$, it must also be the case that $f'(f_1(a), a) = 1$ and $f'(f_2(a), a) = 0$, which is absurd.

We proved that the object \Box is not local by using an idea that can be expressed in general and it is used in the following *Locality Condition* lemma.

Lemma 2.2.6. (Locality Condition -LC) *An object $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$ is local if and only if the following condition holds: For any two n -tuples (f_1, \dots, f_n) and (g_1, \dots, g_n) of local objects of appropriate types and corresponding local associates (f'_1, \dots, f'_n) and (g'_1, \dots, g'_n) , and for any $a : \mathbf{s}$, if, for $i = 1, \dots, n$, $f'_i(a) = g'_i(a)$, then $f(f_1, \dots, f_n, a) = f(g_1, \dots, g_n, a)$.*

Proof. The proof is by induction on the level of the type of f .

(\implies) Let f' be a local associate of the local object $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$. Then, for any $a : \mathbf{s}$ and for any tuples of local objects of appropriate types (f_1, \dots, f_n) and (g_1, \dots, g_n) and local associates (f'_1, \dots, f'_n) and (g'_1, \dots, g'_n) of them, such that for $i = 1, \dots, n$, $f'_i(a) = g'_i(a)$,

$$\begin{aligned} f(f_1, \dots, f_n, a) &= f'(a, f'_1(a), \dots, f'_n(a)) \\ &= f'(a, g'_1(a), \dots, g'_n(a)) = f(g_1, \dots, g_n, a). \end{aligned}$$

(\impliedby) If $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$, we define $f' : \mathbf{s} \times \sigma_1 \times \dots \times \sigma_n \rightarrow \sigma_0$ by

$$f'(a, b_1, \dots, b_n) = \begin{cases} f(f_1, \dots, f_n, a), & \text{if for } i = 1, \dots, n, \text{ there are local} \\ & f_1, \dots, f_n \text{ such that } f'_i(a) = b_i, \\ er_{\sigma_0}, & \text{otherwise.} \end{cases}$$

The function f' is well defined since, by hypothesis, even if for some i , there are two local f_i and g_i and local associates f'_i and g'_i of them respectively, such that $b_i = f'_i(a)$ and $b_i = g'_i(a)$ for some a , it is the case that

$$f(f_1, \dots, f_i, \dots, f_n, a) = f(f_1, \dots, g_i, \dots, f_n, a).$$

The function f' is clearly a local associate of f and thus, f is local. \dashv

In general, the values of the local associates of a local object f may differ only on arguments that are not themselves values of local associates of arguments of f . If we give a specific value to f on these arguments, then we can define a particular, “preferred” local associate of f .

Definition 2.2.7. If $f : \tilde{\sigma}$ is local, a local associate $f_* : \mathbf{s} \rightarrow \sigma$ of it is a *preferred local associate of f* if and only if

- (i) If $f : \tilde{\sigma}_0$, then $f_* = f$.
- (ii) If $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$, then $f_* : \mathbf{s} \times \sigma_1 \times \dots \times \sigma_n \rightarrow \sigma_0$ is such that for any $a : \mathbf{s}$ and any $b_1 : \sigma_1, \dots, b_n : \sigma_n$, if there are no local $f_i : \tilde{\sigma}_i$ with local associates f'_i such that $b_i = f'_i(a)$, then

$$f_*(a, b_1, \dots, b_n) = er_{\sigma_0}.$$

Proposition 2.2.8. If $f : \tilde{\sigma}$ is local, then it has a unique preferred local associate.

Proof. The proof is by induction on the level($\tilde{\sigma}$).

- (i) If $f : \tilde{\sigma}_0$, then $f_* = f$ and trivially, it is unique.
- (ii) Let $f' : \mathbf{s} \times \sigma_1 \times \dots \times \sigma_n \rightarrow \sigma_0$ be a local associate of a local object $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$. We define the function

$$f_*(a, b_1, \dots, b_n) = \begin{cases} f'(a, b_1, \dots, b_n), & \text{if, for } i = 1, \dots, n, \text{ there are local} \\ & f_1, \dots, f_n \text{ such that } f'_i(a) = b_i, \\ er_{\sigma_0}, & \text{otherwise.} \end{cases}$$

The function f_* is a local associate of f since for any appropriately typed local objects f_1, \dots, f_n with local associates f'_1, \dots, f'_n and any $a : \mathbf{s}$,

$$f(f_1, \dots, f_n, a) = f'(a, f'_1(a), \dots, f'_n(a)) = f_*(a, f'_1(a), \dots, f'_n(a)).$$

It also follows immediately from the definition that f_* is a preferred local associate of f .

Suppose now that $f'_* : \mathbf{s} \times \sigma_1 \times \dots \times \sigma_n \rightarrow \sigma_0$ is another preferred local associate of f . We consider cases on the arguments of f_* and f'_* .

For any $a : \mathbf{s}$ and any b_1, \dots, b_n of appropriate types such that there are local f_1, \dots, f_n and local associates of them f'_1, \dots, f'_n , respectively such that $f'_i(a) = b_i$, for $i = 1, \dots, n$,

$$\begin{aligned} f_*(a, b_1, \dots, b_n) &= f'(a, b_1, \dots, b_n) \\ &= f(f_1, \dots, f_n, a) \\ &= f'_*(a, f'_1(a), \dots, f'_n(a)) = f'_*(a, b_1, \dots, b_n), \end{aligned}$$

because f'_* is a local associate of f .

On the other hand, for any $a : \mathbf{s}$ and any other appropriately typed b_1, \dots, b_n ,

$$f_*(a, b_1, \dots, b_n) = er_{\sigma_0} = f'_*(a, b_1, \dots, b_n),$$

because f'_* is a preferred local associate of f . \dashv

Finally, the following theorem summarizes some basic properties of local objects and their local associates.

Theorem 2.2.9. (i) **(Projection)** *If $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_i$ is a projection function such that*

$$f(f_1, \dots, f_n) = f_i,$$

then f is local and the function $f' : \mathbf{s} \times \sigma_1 \times \dots \times \sigma_n \rightarrow \sigma_i$ defined for any $a : \mathbf{s}$ by

$$f'(a, y_1, \dots, y_n) = y_i$$

is a local associate of it.

(ii) **(Evaluation at Local Values)** *If $f : \tilde{\tau}_1 \times \dots \times \tilde{\tau}_n \rightarrow \tilde{\sigma}$ and $y_1 : \tilde{\tau}_1, \dots, y_n : \tilde{\tau}_n$ are local with local associates f' and y'_1, \dots, y'_n respectively, then $f(y_1, \dots, y_n)$ is also local and the function $(f(y_1, \dots, y_n))' : \mathbf{s} \rightarrow \sigma$ defined for any $a : \mathbf{s}$ by*

$$(f(y_1, \dots, y_n))'(a) = f'(a, y'_1(a), \dots, y'_n(a))$$

is a local associate of it.

(iii) **(Composition)** *If $g : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \times \tilde{\sigma} \rightarrow \tilde{\rho}$ and $h : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}$ are local with local associates g' and h' respectively, and*

$$f(x_1, \dots, x_n) = g(x_1, \dots, x_n, h(x_1, \dots, x_n)),$$

then f is also local and the function $f' : \mathbf{s} \times \sigma_1 \times \dots \times \sigma_n \rightarrow \rho$ defined for any $a : \mathbf{s}$ by

$$f'(a, y_1, \dots, y_n) = g'(a, y_1, \dots, y_n, h'(a, y_1, \dots, y_n))$$

is a local associate of it.

(iv) **(λ -abstraction)** *If $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \times \tilde{\tau} \rightarrow \tilde{\rho}$ is local with local associate f' and*

$$g(x_1, \dots, x_n) = \lambda(x) f(x_1, \dots, x_n, x),$$

then g is local and the function $g' : \mathbf{s} \times \sigma_1 \times \dots \times \sigma_n \rightarrow (\tau \rightarrow \rho)$ defined by

$$g'(a, h_1, \dots, h_n) = \lambda(h) f'(a, h_1, \dots, h_n, h)$$

is a local associate of it.

Proof. (i) Suppose $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_i$ is a projection function, so that for appropriately typed f_1, \dots, f_n , $f(f_1, \dots, f_n) = f_i$.

Let $\tilde{\sigma}_i \equiv \tilde{\sigma}_i^1 \times \dots \times \tilde{\sigma}_i^m \rightarrow \tilde{\sigma}_0$. We need to show, by Definition 2.2.2, that for any local $x_1, \dots, x_n, x_{n+1}, \dots, x_{n+m}$ with local associates $x'_1, \dots, x'_n, x'_{n+1}, \dots, x'_{n+m}$ and any $a : \mathbf{s}$,

$$\begin{aligned} f(x_1, \dots, x_n, x_{n+1}, \dots, x_{n+m}, a) \\ = f'(a, x'_1(a), \dots, x'_n(a), x'_{n+1}(a), \dots, x'_{n+m}(a)). \end{aligned}$$

This holds since

$$\begin{aligned} f'(a, x'_1(a), \dots, x'_n(a), x'_{n+1}(a), \dots, x'_{n+m}(a)) \\ = x'_i(a)(x'_{n+1}(a), \dots, x'_{n+m}(a)) \\ = x_i(x_{n+1}, \dots, x_{n+m}, a) \\ = f(x_1, \dots, x_n, x_{n+1}, \dots, x_{n+m}, a). \end{aligned}$$

(ii) Let $\tilde{\sigma} \equiv \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_m \rightarrow \tilde{\sigma}_0$. We need to show, by Definition 2.2.2, that for any local x_1, \dots, x_m of appropriate types with local associates x'_1, \dots, x'_m and any $a : \mathbf{s}$,

$$(f(y_1, \dots, y_n))(x_1, \dots, x_m, a) = (f(y_1, \dots, y_n))'(a, x'_1(a), \dots, x'_m(a)).$$

This holds since

$$\begin{aligned} (f(y_1, \dots, y_n))'(a, x'_1(a), \dots, x'_m(a)) \\ = (f'(a, y'_1(a), \dots, y'_n(a)))(x'_1(a), \dots, x'_m(a)) \\ = f'(a, y'_1(a), \dots, y'_n(a), x'_1(a), \dots, x'_m(a)) \\ = f(y_1, \dots, y_n, x_1, \dots, x_m, a) \\ = (f(y_1, \dots, y_n))(x_1, \dots, x_m, a). \end{aligned}$$

(iii) Suppose that $\tilde{\rho} \equiv \tilde{\rho}_1 \times \dots \times \tilde{\rho}_m \rightarrow \tilde{\rho}_0$. We need to show, by Definition 2.2.2, that for any local $x_1, \dots, x_n, x_{n+1}, \dots, x_{n+m}$ of appropriate types with local associates $x'_1, \dots, x'_n, x'_{n+1}, \dots, x'_{n+m}$ and any $a : \mathbf{s}$,

$$\begin{aligned} f(x_1, \dots, x_n, x_{n+1}, \dots, x_{n+m}, a) \\ = f'(a, x'_1(a), \dots, x'_n(a), x'_{n+1}(a), \dots, x'_{n+m}(a)). \end{aligned}$$

Using case (ii) of this theorem, this holds since

$$\begin{aligned}
& f'(a, x'_1(a), \dots, x'_{n+m}(a)) \\
&= g'(a, x'_1(a), \dots, x'_n(a), h'(a, x'_1(a), \dots, x'_n(a)), x'_{n+1}(a), \dots, x'_{n+m}(a)) \\
&= g'(a, x'_1(a), \dots, x'_n(a), (h(x_1, \dots, x_n))'(a), x'_{n+1}(a), \dots, x'_{n+m}(a)) \\
&= g(x_1, \dots, x_n, h(x_1, \dots, x_n), x_{n+1}, \dots, x_{n+m}, a) \\
&= f(x_1, \dots, x_n, x_{n+1}, \dots, x_{n+m}, a).
\end{aligned}$$

(iv) Suppose that $\tilde{\rho} \equiv \tilde{\rho}_1 \times \dots \times \tilde{\rho}_m \rightarrow \tilde{\rho}_0$. We need to show, by Definition 2.2.2, that for any local $x_1, \dots, x_n, x, x_{n+1}, \dots, x_{n+m}$ of appropriate types with local associates $x'_1, \dots, x'_n, x', x'_{n+1}, \dots, x'_{n+m}$ and any $a : s$,

$$\begin{aligned}
& g(x_1, \dots, x_n, x, x_{n+1}, \dots, x_{n+m}, a) \\
&= g'(a, x'_1(a), \dots, x'_n(a), x'(a), x'_{n+1}(a), \dots, x'_{n+m}(a))
\end{aligned}$$

This holds since

$$\begin{aligned}
& g'(a, x'_1(a), \dots, x'_n(a), x'(a), x'_{n+1}(a), \dots, x'_{n+m}(a)) \\
&= (\lambda(h)f'(a, x'_1(a), \dots, x'_n(a), h))(x'(a), x'_{n+1}(a), \dots, x'_{n+m}(a)) \\
&= f'(a, x'_1(a), \dots, x'_n(a), x'(a), x'_{n+1}(a), \dots, x'_{n+m}(a)) \\
&= f(x_1, \dots, x_n, x, x_{n+1}, \dots, x_{n+m}, a) \\
&= g(x_1, \dots, x_n, x, x_{n+1}, \dots, x_{n+m}, a). \quad \dashv
\end{aligned}$$

The section ends with a simple straightforward corollary of the λ -abstraction case of the theorem.

Corollary 2.2.10. *If $f : \tilde{\tau}$ is local with local associate f' and for $x_i : \tilde{\sigma}_i$,*

$$g = \lambda(x_1) \dots \lambda(x_n)f,$$

then g is local and the function $g' : s \times \sigma_1 \times \dots \times \sigma_n \rightarrow \tau$ defined for any $a : s$ by

$$g'(a) = \lambda(z_1) \dots \lambda(z_n)(f'(a))$$

is a local associate of it.

2.3 Locality Indices of Type $\tilde{\sigma}$

Local objects are only some of the objects that interpret natural language constants. Like \square , there are natural examples of objects that are not local

and they may even exhibit different locality behavior at every one of their arguments.

The basic characteristic of the approach towards locality that is adopted here is that *there is no assumption of a uniform, local or non local, behavior of objects or even of the arguments of a single object*. As a consequence, there is a need of a suitable description of this behavior which will be formalized in a way similar to that of local objects in Section 2.2.

Example 2.3.1. Consider the function $former : (\tilde{e} \rightarrow \tilde{t}) \times \tilde{e} \rightarrow \tilde{t}$ which interprets the constant *former*. For any objects $f_1 : \tilde{e} \rightarrow \tilde{t}$ and $f_2 : \tilde{e}$ and any $a : s$, the function is defined by

$$former(f_1, f_2, a) = 1 \iff f_2(a) \text{ has property } f_1 \text{ at some state } b, \\ \text{prior in time to } a$$

For example, the sentence ‘John is a former minister’ is rendered in L_{ar}^λ by the term $former(minister, John)$. Its denotation at some state $a : s$ depends on the denotation of *minister* at other states that indicate a moment of time in the past with respect to a while in the case of the denotation of *John*, only its value at state a is needed.

The type $(\tilde{e} \rightarrow \tilde{t}) \times \tilde{e} \rightarrow \tilde{t}$ of *former* expresses formally that it expects a first argument of type $(\tilde{e} \rightarrow \tilde{t})$ and a second one of \tilde{e} . We define in what follows a corresponding formalization for each type, a *closed locality index of type $\tilde{\sigma}$* , such that, for example, in the case of *former*, it expresses the fact that any argument of type $(\tilde{e} \rightarrow \tilde{t})$ is used by *former* non locally, noted by the digit 1, while any argument of type \tilde{e} is used locally, noted by 0. The index for *former* expresses also how any of its first arguments treats its own arguments of type \tilde{e} — it describes thus totally the locality behavior of the object.

Definition 2.3.2. (Closed Locality Index of Type $\tilde{\sigma}$) A *closed locality index of type $\tilde{\sigma}$* is any sequence of the form

$$\ell \mapsto t$$

of two independent parts: a *closed locality input index ℓ of type $\tilde{\sigma}$* and a *closed locality output index t* .

A closed locality output index t is an *index constant* defined by

$$t ::= 0 \mid 1.$$

\tilde{t}, \tilde{e}	l
$\tilde{t} \rightarrow \tilde{t}$	$\langle l \hookrightarrow 0 \rangle, \langle l \hookrightarrow 1 \rangle$
$\tilde{e} \times \tilde{e} \rightarrow \tilde{t}$	$\langle l \hookrightarrow 0, l \hookrightarrow 0 \rangle, \langle l \hookrightarrow 1, l \hookrightarrow 1 \rangle,$ $\langle l \hookrightarrow 1, l \hookrightarrow 0 \rangle, \langle l \hookrightarrow 0, l \hookrightarrow 1 \rangle$
$(\tilde{e} \rightarrow \tilde{t}) \times \tilde{e} \rightarrow \tilde{t}$	$\langle \langle l \hookrightarrow 0 \rangle \hookrightarrow 0, l \hookrightarrow 0 \rangle, \langle \langle l \hookrightarrow 1 \rangle \hookrightarrow 1, l \hookrightarrow 1 \rangle$ $\langle \langle l \hookrightarrow 1 \rangle \hookrightarrow 0, l \hookrightarrow 0 \rangle, \langle \langle l \hookrightarrow 0 \rangle \hookrightarrow 1, l \hookrightarrow 0 \rangle$ $\langle \langle l \hookrightarrow 0 \rangle \hookrightarrow 0, l \hookrightarrow 1 \rangle, \langle \langle l \hookrightarrow 1 \rangle \hookrightarrow 1, l \hookrightarrow 0 \rangle$ $\langle \langle l \hookrightarrow 1 \rangle \hookrightarrow 0, l \hookrightarrow 1 \rangle, \langle \langle l \hookrightarrow 0 \rangle \hookrightarrow 1, l \hookrightarrow 1 \rangle$

Table 2.1: Examples of closed locality input indices.

A closed locality input index ℓ of type $\tilde{\sigma}$ is defined by recursion on $\text{level}(\tilde{\sigma})$ as follows:

If $\tilde{\sigma} \equiv \tilde{\sigma}_0$, $\ell := l$, where l is a fixed constant symbol.

If $\tilde{\sigma} \equiv \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$, a closed locality input index ℓ of $\tilde{\sigma}$ is an expression

$$\ell := \langle \ell_1 \hookrightarrow t_1, \dots, \ell_n \hookrightarrow t_n \rangle$$

where ℓ_1, \dots, ℓ_n are closed locality input indices of types $\tilde{\sigma}_1, \dots, \tilde{\sigma}_n$ respectively and t_1, \dots, t_n are index constants.

In Table 2.1, all the possible closed locality input indices of some low level types are shown.

Notice also that, in general, by our understanding of tuples,

$$\langle \ell_1 \hookrightarrow t_1, \dots, \ell_n \hookrightarrow t_n \rangle \hookrightarrow t \equiv \langle \ell_1 \hookrightarrow t_1, \langle \ell_2 \hookrightarrow t_2, \dots, \ell_n \hookrightarrow t_n \rangle \rangle \hookrightarrow t.$$

Finally, there are some closed indices of special interest for any type.

Definition 2.3.3. If all the index constants of a closed locality (input) index of any type are equal to 1 then it is called *standard* while if all its index constants are equal to 0, it is called *local*.

Now, it is natural to generalize the definition of a closed locality index with the use of binary variables.

Definition 2.3.4. A *locality index of type $\tilde{\sigma}$* is defined as in Definition 2.3.2 where a locality output index t is now an *index token* defined by

$$t := 0 \mid 1 \mid b$$

where b is an *index variable*.

It follows that if every index variable b occurring anywhere in a locality index ℓ of type $\tilde{\sigma}$ is replaced by 0 or 1, the resulting string is a closed locality index of the same type. Formally,

Definition 2.3.5. A *substitution* is a function

$$\rho : \{\text{Index Variables}\} \rightarrow \{0, 1\} \cup \{\text{Index Variables}\}.$$

We simply extend $\rho(\ell)$ by replacing all the index variables b that occur in a locality input index ℓ by $\rho(b)$. An *evaluation* ρ' is analogously a special case of a substitution such that

$$\rho' : \{\text{Index Variables}\} \rightarrow \{0, 1\}.$$

For every type $\tilde{\sigma}$, there is locality index with index variables such that any locality index of that type can be obtained by a substitution of the index variables that occur in it.

Lemma 2.3.6. (Generic Locality Index) *For each type $\tilde{\sigma}$, there is a locality (input) index, called generic, such that all its index tokens are variables and no index variable occurs more than once. Moreover:*

- (i) *Any locality (input) index of type $\tilde{\sigma}$ can be obtained from it by an appropriate substitution of its index variables.*
- (ii) *It is unique up to alphabetic variance of the index variables that occur in it.*
- (iii) *Any other locality (input) index of type $\tilde{\sigma}$ that has property (i) is an alphabetic variant of it.*

Proof. The proof is by induction on the level($\tilde{\sigma}$).

If $\tilde{\sigma} \equiv \tilde{\sigma}_0$, then the generic locality index is $\ell \mapsto b$ where b is any index variable.

If $\tilde{\sigma} \equiv \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$, let ℓ_1, \dots, ℓ_n be generic locality input indices of $\tilde{\sigma}_1, \dots, \tilde{\sigma}_n$ respectively with distinct index variables. The generic locality index of $\tilde{\sigma}$ is

$$\ell \equiv \langle \ell_1 \mapsto b_1, \dots, \ell_n \mapsto b_n \rangle \mapsto b$$

where b_1, \dots, b_n, b are distinct fresh index variables.

For property (i), it is enough to consider for each particular locality index $\langle \ell'_1 \mapsto t_1, \dots, \ell'_n \mapsto t_n \rangle \mapsto t$ of $\tilde{\sigma}$, a substitution ρ such that $\rho(\ell_i) = \ell'_i$, $\rho(b_i) = t_i$ and $\rho(b) = t$ ($i = 1, \dots, n$). Since each index variable occurs once, ρ is well defined.

In order to prove property (ii), let $\ell' \equiv \langle \ell'_1 \hookrightarrow b'_1, \dots, \ell'_n \hookrightarrow b'_n \rangle \hookrightarrow b'$ be another generic locality index of type $\tilde{\sigma}$. It is straightforward that the substitution ρ such that $\rho(\ell') = \ell$ is an alphabetic renaming of the index variables of ℓ .

To prove property (iii), let $\ell' \equiv \langle \ell'_1 \hookrightarrow t'_1, \dots, \ell'_n \hookrightarrow t'_n \rangle \hookrightarrow t'$ be any locality index of type $\tilde{\sigma}$ such that property (i) holds. Then, there is a substitution ρ such that $\rho(\ell') = \ell$. Now, since (i) holds for ℓ , there is a substitution ρ' such that $\rho'(\ell) = \ell'$. Thus, it must be the case that $\rho(\rho'(\ell)) = \ell$. So, ρ is the inverse of ρ' and thus, ℓ' is an alphabetic variant of ℓ . \dashv

2.4 Associate of an Object with Respect to a Closed Locality Input Index of Its Type

Suppose that $\langle l \hookrightarrow b_1, l \hookrightarrow b_2 \rangle$ is the generic locality input index of an object $f : \tilde{\mathbf{e}} \times \tilde{\mathbf{e}} \rightarrow \tilde{\mathbf{t}}$. In Table 2.1, it was shown that there are four possible closed locality input indices for this object. As we have mentioned before, the two different index constants, 0 and 1, symbolize the local and non local use respectively of the corresponding argument that each one accompanies.

For example, the object f , considered with respect to the closed locality input index $\langle l \hookrightarrow 0, l \hookrightarrow 1 \rangle$, uses its first argument with locality input index l (since it is of level one type) locally and the second one non locally. Each of the other three possible closed locality input indices of f expresses a different behavior of f with respect to the way its computation at a particular state a uses the values of its arguments on other states than a . If $f = \textit{love}$, then the locality input index that we should use to express its locality behavior as described in Example 2.2.1, is of course $\langle l \hookrightarrow 0, l \hookrightarrow 0 \rangle$.

Locality input indices are more complex if we consider objects of type of level greater than two. The generic locality input index of type $(\tilde{\mathbf{e}} \rightarrow \tilde{\mathbf{t}}) \rightarrow \tilde{\mathbf{t}}$ is $\langle \langle l \hookrightarrow b_2 \rangle \hookrightarrow b_1 \rangle$ and one of its closed instances is, $\langle \langle l \hookrightarrow 0 \rangle \hookrightarrow 1 \rangle$. This index indicates that an object of this type uses its first (and only) argument non locally but only when the argument itself uses its own argument locally. Another closed instance of its generic locality input index is $\langle \langle l \hookrightarrow 1 \rangle \hookrightarrow 1 \rangle$ which expresses the non local use of the object's first argument when, in contrast, the argument itself uses its own argument non locally.

Locality input indices describe the locality behavior of objects and of their arguments without assuming any fixed treatment of any kind. All the different combinations are allowed in an attempt to understand locality as well as possible. In Section 2.2, we defined local objects with the use of local associates and now, a generalized notion of an *associate* of an object will be

\tilde{e}	l	$s \rightarrow e$
$\tilde{e} \times \tilde{e} \rightarrow \tilde{t}$	$\langle l \mapsto 0, l \mapsto 0 \rangle$	$s \rightarrow (e \times e \rightarrow t)$
	$\langle l \mapsto 1, l \mapsto 1 \rangle$	$s \rightarrow (\tilde{e} \times \tilde{e} \rightarrow t)$
	$\langle l \mapsto 0, l \mapsto 1 \rangle$	$s \rightarrow (e \times \tilde{e} \rightarrow t)$
$(\tilde{e} \rightarrow \tilde{t}) \times \tilde{e} \rightarrow \tilde{t}$	$\langle \langle l \mapsto 0 \rangle \mapsto 0, l \mapsto 0 \rangle$	$s \rightarrow ((e \rightarrow t) \times e \rightarrow t)$
	$\langle \langle l \mapsto 1 \rangle \mapsto 1, l \mapsto 1 \rangle$	$s \rightarrow ((s \rightarrow (\tilde{e} \rightarrow t)) \times \tilde{e} \rightarrow t)$
	$\langle \langle l \mapsto 0 \rangle \mapsto 1, l \mapsto 0 \rangle$	$s \rightarrow ((s \rightarrow (e \rightarrow t)) \times e \rightarrow t)$

Table 2.2: Examples of associate types.

defined with respect to a closed locality input index of its type. First, we will need a simple transformation that associates with each state-dependent type $\tilde{\sigma}$ and each closed locality input index ℓ the *associate type* $(\tilde{\sigma})_*^\ell$ which is going to be the type of the value of the corresponding associate at any state.

Definition 2.4.1. For each type $\tilde{\sigma}$ and each closed locality input index ℓ of type $\tilde{\sigma}$, the *associate type* $(\tilde{\sigma})_*^\ell$ is defined by recursion on $\text{level}(\tilde{\sigma})$ as follows:

$$\begin{aligned}
(\tilde{\sigma}_0)_*^\ell &:= \sigma_0 \\
(\tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0)_*^{\langle \ell_1 \mapsto t_1, \dots, \ell_n \mapsto t_n \rangle} &:= \tau_1 \times \dots \times \tau_n \rightarrow \sigma_0, \\
\text{where } \tau_i &:= \begin{cases} (\tilde{\sigma}_i)_*^{\ell_i}, & \text{if } t_i \equiv 0 \\ s \rightarrow (\tilde{\sigma}_i)_*^{\ell_i}, & \text{if } t_i \equiv 1. \end{cases}
\end{aligned}$$

In Table 2.2, we present some types and their corresponding types of the form $(s \rightarrow \text{associate type})$ with respect to specific closed locality input indices. Notice that each time if the closed locality input index ℓ is the local one, then

$$(\tilde{\sigma})_*^\ell \equiv \sigma.$$

On the other hand, for the standard locality input index of any type, we can easily prove the following proposition.

Proposition 2.4.2. For any type $\tilde{\sigma}$, if ℓ is its standard closed locality input index, and

$$\text{Stand}(\tilde{\sigma}) := s \rightarrow (\tilde{\sigma})_*^\ell,$$

then

$$\text{Stand}(\tilde{\sigma}_0) := \tilde{\sigma}_0$$

$$\text{Stand}(\tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0) := \mathbf{s} \times \text{Stand}(\tilde{\sigma}_1) \times \dots \times \text{Stand}(\tilde{\sigma}_n) \rightarrow \sigma_0.$$

For example, consider the associate types with respect to the standard indices that are shown in Table 2.2.

Definition 2.4.3. (Associate of an Object with respect to a Closed Input Index of its Type) For each $f : \tilde{\sigma}$ and any closed locality input index ℓ of its type, we define what it means for

$$f' : \mathbf{s} \rightarrow (\tilde{\sigma})_*^\ell$$

to be an *associate of f with respect to ℓ* by recursion on the level of $\tilde{\sigma}$.

(i) If $f : \tilde{\sigma}_0$ and $\ell \equiv l$, then $f' : \mathbf{s} \rightarrow \sigma_0$ is an associate of f if and only if $f' = f$.

(ii) If $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$ and $\ell \equiv \langle \ell_1 \hookrightarrow t_1, \dots, \ell_n \hookrightarrow t_n \rangle$, then $f' : \mathbf{s} \rightarrow (\tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0)_*^\ell$ is an associate of f w.r.t. ℓ if and only if for any $f_i : \tilde{\sigma}_i$ and any $a : \mathbf{s}$,

$$f(f_1, \dots, f_n, a) = f'(a, F_1, \dots, F_n),$$

where, for $i = 1, \dots, n$,

$$F_i := \begin{cases} f'_i(a), & \text{if } t_i \equiv 0, \\ f'_i, & \text{if } t_i \equiv 1 \end{cases}$$

and $f'_i : \mathbf{s} \rightarrow (\tilde{\sigma}_i)_*^{\ell_i}$ is any associate of f_i with respect to ℓ_i .

A closed locality input index ℓ of type $\tilde{\sigma}$ is a *closed locality input index* of an object $f : \tilde{\sigma}$ if and only if there is an associate of f with respect to ℓ .

Thus, by this definition, an object $f : \tilde{\sigma}$ is local if and only if it has an associate with respect to the local closed locality input index of $\tilde{\sigma}$.

Now, the associates of an object with respect to the same closed locality input index are related as in the local case.

Proposition 2.4.4. *Suppose that $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$ and f_1 and f_2 are associates of it with respect to $\ell \equiv \langle \ell_1 \hookrightarrow t_1, \dots, \ell_n \hookrightarrow t_n \rangle$. Then, for any objects f_1, \dots, f_n such that for $i = 1, \dots, n$, f'_i is an associate of f_i with respect to ℓ_i and any $a : \mathbf{s}$,*

$$f_1(a, F_1, \dots, F_n) = f_2(a, F_1, \dots, F_n),$$

where

$$F_i = \begin{cases} f'_i(a), & \text{if } t_i \equiv 0, \\ f'_i, & \text{if } t_i \equiv 1. \end{cases}$$

To check whether an object has an associate with respect to an arbitrary closed locality input index of its type, a generalized version of the LC Lemma 2.2.6 is used.

Lemma 2.4.5. (General Locality Condition - GLC) *An object $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$ has an associate w.r.t. $\ell \equiv \langle \ell_1 \vdash t_1, \dots, \ell_n \vdash t_n \rangle$ if and only if the following condition holds: For any two n -tuples (f_1, \dots, f_n) and (g_1, \dots, g_n) and corresponding associates f'_1, \dots, f'_n and g'_1, \dots, g'_n such that for $i = 1, \dots, n$, f'_i and g'_i are associates of f_i and g_i with respect to ℓ_i , respectively, and for any $a : s$, if for $i = 1, \dots, n$, either $f'_i(a) = g'_i(a)$ (if $t_i \equiv 0$) or $f'_i = g'_i$ (if $t_i \equiv 1$), then $f(f_1, \dots, f_n, a) = f(g_1, \dots, g_n, a)$.*

Proof. The proof is by induction on the level of the type of f .

(\implies) Let $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$ have an associate with respect to $\ell \equiv \langle \ell_1 \vdash t_1, \dots, \ell_n \vdash t_n \rangle$. Suppose that for $i = 1, \dots, n$, f_i is an object that has an associate f'_i with respect to ℓ_i , then for any $a : s$

$$F_i = \begin{cases} f'_i(a), & \text{if } t_i \equiv 0, \\ f'_i, & \text{if } t_i \equiv 1. \end{cases}$$

We define analogously G_i for $i = 1, \dots, n$. If f' is an associate of f with respect to ℓ , then, for any $a : s$ and for any tuples of objects of appropriate types (f_1, \dots, f_n) and (g_1, \dots, g_n) such that for $i = 1, \dots, n$, f_i and g_i have associates with respect to ℓ_i ,

$$\begin{aligned} f(f_1, \dots, f_n, a) &= f'(a, F_1, \dots, F_n) \\ &= f'(a, G_1, \dots, G_n) = f(g_1, \dots, g_n, a). \end{aligned}$$

(\impliedby) If $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$ and $\ell \equiv \langle \ell_1 \vdash t_1, \dots, \ell_n \vdash t_n \rangle$, then consider the function $f' : s \rightarrow (\tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0)_{*}^{\ell}$, defined as

$$f'(a, b_1, \dots, b_n) = \begin{cases} f(f_1, \dots, f_n, a), & \text{if for } i = 1, \dots, n, \text{ there are} \\ & f_1, \dots, f_n \text{ with corresponding} \\ & \text{associates such that } b_i = F_i, \\ er_{\sigma_0}, & \text{otherwise.} \end{cases}$$

The function f' is well defined since, by hypothesis, even if for some i , there are two different f_i and g_i such that $b_i = F_i$ and $b_i = G_i$ for some a , then

$$f(f_1, \dots, f_i, \dots, f_n, a) = f(f_1, \dots, g_i, \dots, f_n, a).$$

Trivially, f' is an associate of f with respect to ℓ . \dashv

For the standard locality input index of an object, we do not need to use GLC Lemma since the following holds.

Proposition 2.4.6. (i) *For every $f : \tilde{\sigma}$, there is a unique associate of f w.r.t. the standard locality input index of its type.*

(ii) *For every $f' : s \rightarrow (\tilde{\sigma})_*^\ell$ where ℓ is the standard locality input index of $\tilde{\sigma}$, there is a unique f such that f' is an associate of f w.r.t. ℓ .*

Proof. The proof is by induction on the level($\tilde{\sigma}$).

If $\tilde{\sigma} \equiv \tilde{\sigma}_0$, both claims are trivial.

Let $\tilde{\sigma} \equiv \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$ and $\ell \equiv \langle \ell_1 \hookrightarrow 1, \dots, \ell_n \hookrightarrow 1 \rangle$ be its standard locality input index. Notice that for $i = 1, \dots, n$, ℓ_i is the standard locality input index of type $\tilde{\sigma}_i$ respectively.

Suppose that for $i = 1, \dots, n$, both claims hold. That is, for every $f_i : \tilde{\sigma}_i$, there is a unique associate f'_i with respect to ℓ_i and for every $f'_i : \text{Stand}(\tilde{\sigma}_i)$ there is a unique f_i such that f'_i is the associate of f_i w.r.t. ℓ_i .

(i) We define $f' : s \times \text{Stand}(\tilde{\sigma}_1) \times \dots \times \text{Stand}(\tilde{\sigma}_n) \rightarrow \sigma_0$ for appropriate typed f'_1, \dots, f'_n by

$$f'(a, f'_1, \dots, f'_n) = f(f_1, \dots, f_n, a)$$

such that for $i = 1, \dots, n$, f_i is the unique object such that f'_i is the associate of f_i w.r.t. ℓ_i .

The function f' is clearly an associate of f w.r.t. ℓ .

Suppose now that there is another associate g of f w.r.t. ℓ and that there is a n -tuple (f'_1, \dots, f'_n) such that

$$f'(a, f'_1, \dots, f'_n) \neq g(a, f'_1, \dots, f'_n).$$

Thus, it must be the case that

$$g(a, f'_1, \dots, f'_n) \neq f(f_1, \dots, f_n, a)$$

such that for $i = 1, \dots, n$, f_i is the unique object such that f'_i is its associate w.r.t. ℓ_i . Since for each f_i there is a unique associate w.r.t. ℓ_i and g is an associate of f w.r.t. ℓ , we have a contradiction.

(ii) We define $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$ for appropriately typed f_1, \dots, f_n and their unique associates f'_1, \dots, f'_n with respect to ℓ_1, \dots, ℓ_n respectively, by

$$f(f_1, \dots, f_n, a) = f'(a, f'_1, \dots, f'_n).$$

f is well defined and f' is an associate of f with respect to ℓ .

Suppose that there is another object g such that f' is an associate of it with respect to ℓ . Then, for any f_1, \dots, f_n and their associates f'_1, \dots, f'_n with respect to ℓ_1, \dots, ℓ_n respectively

$$g(f_1, \dots, f_n, a) = f'(a, f'_1, \dots, f'_n) = f(f_1, \dots, f_n, a). \quad \dashv$$

Next, we generalize the notion of *preferred associate* of an object with respect to a closed locality input index and we prove that it is unique.

Proposition 2.4.7. *If $f : \tilde{\sigma}$ has an associate with respect to a closed locality input index ℓ of its type, then it has a unique associate f_*^ℓ (called preferred associate with respect to ℓ) such that:*

- (i) *If $f : \tilde{\sigma}_0$, then $f_*^\ell = f$.*
- (ii) *If $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$ and $\ell \equiv \langle \ell_1 \hookrightarrow t_1, \dots, \ell_n \hookrightarrow t_n \rangle$, then $f_*^\ell : \mathbf{s} \rightarrow (\tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0)_*^\ell$ is an associate of f with respect to ℓ such that for any $a : \mathbf{s}$ and any appropriately typed b_1, \dots, b_n , if they are such that for $i = 1, \dots, n$, there are no $f_i : \tilde{\sigma}_i$ with associates f'_i with respect to ℓ_i such that*

$$b_i = \begin{cases} f'_i(a), & \text{if } t_i \equiv 0, \\ f'_i, & \text{if } t_i \equiv 1, \end{cases}$$

then

$$f_*^\ell(a, b_1, \dots, b_n) = er_{\sigma_0}.$$

Proof. If $f : \tilde{\sigma}_0$, then it holds trivially.

Let f' be an associate of $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$ with respect to a closed locality input index $\ell \equiv \langle \ell_1 \hookrightarrow t_1, \dots, \ell_n \hookrightarrow t_n \rangle$. Suppose that for $i = 1, \dots, n$, f'_i is an associate of an object $f_i : \tilde{\sigma}_i$ with respect to the closed locality input index ℓ_i and F_i is, for any $a : \mathbf{s}$

$$F_i = \begin{cases} f'_i(a), & \text{if } t_i \equiv 0, \\ f'_i, & \text{if } t_i \equiv 1. \end{cases}$$

We define f_*^ℓ by

$$f_*^\ell(a, b_1, \dots, b_n) = \begin{cases} f'(a, b_1, \dots, b_n), & \text{if, for } i = 1, \dots, n, \text{ there are } f_i \text{ such} \\ & \text{that } b_i = F_i, \\ er_{\sigma_0}, & \text{otherwise.} \end{cases}$$

The function f_*^ℓ is trivially an associate of f with respect to ℓ since for appropriately typed f_1, \dots, f_n with corresponding associates and any $a : \mathbf{s}$,

$$f(f_1, \dots, f_n, a) = f'(a, F_1, \dots, F_n) = f_*^\ell(a, F_1, \dots, F_n).$$

By its definition, f_*^ℓ is also a preferred one.

Now, suppose that g_*^ℓ is another preferred associate of f with respect to ℓ . It is either the case that

$$f_*^\ell(a, b_1, \dots, b_n) = er_{\sigma_0} = g_*^\ell(a, b_1, \dots, b_n),$$

or,

$$\begin{aligned} f_*^\ell(a, b_1, \dots, b_n) &= f'(a, b_1, \dots, b_n) \\ &= f(f_1, \dots, f_n, a) = g_*^\ell(a, b_1, \dots, b_n). \end{aligned} \quad \dashv$$

Finally, the properties of associates are described in the next theorem that generalizes Theorem 2.2.9.

Theorem 2.4.8. (i) (**Projection**) *If $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_i$ is such that*

$$f(f_1, \dots, f_n) = f_i,$$

and $\ell \equiv \langle \ell_1 \mapsto t_1, \dots, \ell_i \mapsto t_i, \dots, \ell_n \mapsto t_n, \ell_i \rangle$ is any closed locality input index of its type, then f' defined for any $a : \mathbf{s}$ by

$$f'(a, z_1, \dots, z_n) = \begin{cases} z_i, & \text{if } t_i \equiv 0 \\ z_i(a), & \text{if } t_i \equiv 1 \end{cases}$$

is an associate of f with respect to ℓ .

(ii) (**Evaluation**) *If f' is an associate of $f : \tilde{\tau}_1 \times \dots \times \tilde{\tau}_k \rightarrow \tilde{\sigma}$ w.r.t. $\langle m_1 \mapsto s_1, \dots, m_k \mapsto s_k, \ell \rangle$ and for $i = 1, \dots, k$, y'_i is an associate of $y_i : \tilde{\tau}_i$ w.r.t. m_i , then the function $(f(y_1, \dots, y_k))'$ defined for any $a : \mathbf{s}$ by*

$$(f(y_1, \dots, y_k))'(a) = f'(a, Y_1, \dots, Y_k)$$

where

$$Y_i = \begin{cases} y'_i(a), & \text{if } s_i \equiv 0, \\ y'_i, & \text{if } s_i \equiv 1, \end{cases}$$

is an associate of $f(y_1, \dots, y_k)$ w.r.t. ℓ .

(iii) (**Composition**) *Suppose that*

$$f(x_1, \dots, x_k) = g(x_1, \dots, x_k, h(x_1, \dots, x_k)),$$

where $g : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_k \times \tilde{\sigma} \rightarrow \tilde{\rho}$ and $h : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_k \rightarrow \tilde{\sigma}$ and that g' is an associate of g w.r.t. $\langle m_1 \mapsto s_1, \dots, m_k \mapsto s_k, \ell_0 \mapsto t_0, \ell \rangle$ and h' is an associate of h w.r.t. $\langle m_1 \mapsto s_1, \dots, m_k \mapsto s_k, \ell_0 \rangle$; assume that for $j = 1, \dots, k$, $t_0 \leq s_j$, and let the function f' be defined for any $a : s$ by

$$f'(a, z_1, \dots, z_k) = g'(a, z_1, \dots, z_k, H(z_1, \dots, z_k)),$$

where

$$H(z_1, \dots, z_k) = \begin{cases} h'(a, z_1, \dots, z_k), & \text{if } t_0 \equiv 0 \\ (b \mapsto h'(b, z_1, \dots, z_k)), & \text{if } t_0 \equiv 1. \end{cases}$$

Then, f' is an associate of f with respect to $\langle m_1 \mapsto s_1, \dots, m_k \mapsto s_k, \ell \rangle$.

(iv) (**λ -abstraction**) If f' is an associate of $f : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \times \tilde{\sigma} \rightarrow \tilde{\rho}$ with respect to $\langle m_1 \mapsto s_1, \dots, m_n \mapsto s_n, m \mapsto s, \ell \rangle$ and

$$g(x_1, \dots, x_n) = \lambda(x)f(x_1, \dots, x_n, x),$$

then the function g' defined by

$$g'(a, z_1, \dots, z_n) = \lambda(z)f'(a, z_1, \dots, z_n, z)$$

is an associate of g with respect to $\langle m_1 \mapsto s_1, \dots, m_n \mapsto s_n, m \mapsto s, \ell \rangle$.

Proof. (i) Notice that for $j = 1, \dots, n$, if $t_j \equiv 0$, then $z_j : (\tilde{\sigma}_j)_*^{\ell_j}$ while if $t_j \equiv 1$, $z_j : \mathbf{s} \rightarrow (\tilde{\sigma}_j)_*^{\ell_j}$. Now, let $\ell_i \equiv \langle \ell_{n+1} \mapsto t_{n+1}, \dots, \ell_{n+k} \mapsto t_{n+k} \rangle$.

Suppose that $x_1, \dots, x_i, \dots, x_n, x_{n+1}, \dots, x_{n+k}$ are objects of appropriate types and x'_1, \dots, x'_{n+k} are associates of them with respect to $\ell_1, \dots, \ell_{n+k}$, respectively. Suppose that for $j = 1, \dots, n+k$,

$$X_j = \begin{cases} x'_j(a), & \text{if } t_j \equiv 0 \\ x'_j, & \text{if } t_j \equiv 1 \end{cases}$$

We need to show that

$$f(x_1, \dots, x_{n+k}, a) = f'(a, X_1, \dots, X_{n+k}).$$

If $t_i \equiv 0$, then this holds since

$$\begin{aligned} f'(a, X_1, \dots, x'_i(a), \dots, X_n, X_{n+1}, \dots, X_{n+k}) &= x'_i(a)(X_{n+1}, \dots, X_{n+k}) \\ &= x_i(x_{n+1}, \dots, x_{n+k}, a) \\ &= f(x_1, \dots, x_i, \dots, x_{n+k}). \end{aligned}$$

If $t_i \equiv 1$, similarly,

$$\begin{aligned} f'(a, X_1, \dots, x'_i, \dots, X_n, X_{n+1}, \dots, X_{n+k}) &= x'_i(a)(X_{n+1}, \dots, X_{n+k}) \\ &= x_i(x_{n+1}, \dots, x_{n+k}, a) \\ &= f(x_1, \dots, x_i, \dots, x_{n+k}). \end{aligned}$$

(ii) Let $\tilde{\sigma} \equiv \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$ and $\ell \equiv \langle \ell_1 \mapsto t_1, \dots, \ell_n \mapsto t_n \rangle$.

For any objects x_1, \dots, x_n of appropriate types with associates x'_1, \dots, x'_n with respect to ℓ_1, \dots, ℓ_n respectively, and any $a : \mathbf{s}$, if, for any $i = 1, \dots, n$,

$$X_i = \begin{cases} x'_i(a), & \text{if } t_i \equiv 0, \\ x'_i, & \text{if } t_i \equiv 1, \end{cases}$$

then,

$$\begin{aligned} (f(y_1, \dots, y_k))(x_1, \dots, x_n, a) &= f(y_1, \dots, y_k, x_1, \dots, x_n, a) \\ &= f'(a, Y_1, \dots, Y_k, X_1, \dots, X_n) \\ &= (f(y_1, \dots, y_k))'(a, X_1, \dots, X_n). \end{aligned}$$

By Definition 2.4.3, $(f(y_1, \dots, y_k))'$ is an associate of $f(y_1, \dots, y_k)$ with respect to ℓ .

(iii) Notice that for $j = 1, \dots, k$, if $s_j \equiv 0$, then $z_j : (\tilde{\sigma}_j)_*^{m_j}$ while if $s_j \equiv 1$, $z_j : \mathbf{s} \rightarrow (\tilde{\sigma}_j)_*^{m_j}$.

Now, let $\tilde{\rho} \equiv \tilde{\rho}_1 \times \dots \times \tilde{\rho}_n \rightarrow \tilde{\rho}_0$ and $\ell \equiv \langle \ell_1 \mapsto t_1, \dots, \ell_n \mapsto t_n \rangle$.

Let also $h_1, \dots, h_k, x_1, \dots, x_n$ be any objects of appropriate types with associates $h'_1, \dots, h'_k, x'_1, \dots, x'_n$ with respect to $m_1, \dots, m_k, \ell_1, \dots, \ell_n$ respectively and let for $i = 1, \dots, n$ and $j = 1, \dots, k$

$$X_i = \begin{cases} x'_i(a), & \text{if } t_i \equiv 0, \\ x'_i, & \text{if } t_i \equiv 1 \end{cases} \text{ and } H_j = \begin{cases} h'_j(a), & \text{if } s_j \equiv 0, \\ h'_j, & \text{if } s_j \equiv 1. \end{cases}$$

If $t_0 \equiv 0$, then, for any $a : \mathbf{s}$, using case (ii),

$$\begin{aligned} f'(a, H_1, \dots, H_k, X_1, \dots, X_n) &= g'(a, H(H_1, \dots, H_k), X_1, \dots, X_n) \\ &= g'(a, h'(a, H_1, \dots, H_k), X_1, \dots, X_n) \\ &= g'(a, (h(h_1, \dots, h_k))'(a), X_1, \dots, X_n) \\ &= g(h_1, \dots, h_k, (h(h_1, \dots, h_k)), x_1, \dots, x_n, a) \\ &= f(h_1, \dots, h_k, x_1, \dots, x_n, a). \end{aligned}$$

If $t_0 \equiv 1$, then for $j = 1, \dots, k$, $s_j \equiv 1$, and using again case (ii),

$$\begin{aligned}
 f'(a, h'_1, \dots, h'_k, X_1, \dots, X_n) &= g'(a, h'_1, \dots, h'_k, b \mapsto h'(b, h'_1, \dots, h'_k), X_1, \dots, X_n) \\
 &= g'(a, h'_1, \dots, h'_k, b \mapsto (h(h_1, \dots, h_k))'(b), X_1, \dots, X_n) \\
 &= g'(a, h'_1, \dots, h'_k, (h(h_1, \dots, h_k))', X_1, \dots, X_n) \\
 &= g(h_1, \dots, h_k, h(h_1, \dots, h_k), x_1, \dots, x_n, a) \\
 &= f(h_1, \dots, h_k, x_1, \dots, x_n, a).
 \end{aligned}$$

In both cases, by Definition 2.4.3, f' is an associate of f with respect to $\langle m_1 \mapsto s_1, \dots, m_k \mapsto s_k, \ell \rangle$.

Notice that the restriction $t_0 \leq s_j$ is necessary in order to define f' in the case where $t_0 \equiv 1$. Intuitively, f' is defined at a by the composition of g' and h' and if $t_0 \equiv 1$, it needs to compute the associate $(h(z_1, \dots, z_k))'$ and not just its value at a . Thus, for each z_j , it needs its associate and not just the value of its associate at a .

(iv) Notice that for $j = 1, \dots, n$, if $s_j \equiv 0$, then $z_j : (\tilde{\sigma}_j)^{m_j}_*$ while if $s_j \equiv 1$, $z_j : \mathbf{s} \rightarrow (\tilde{\sigma}_j)^{m_j}_*$, and if $s \equiv 0$, then $z : (\tilde{\sigma})^m_*$ while if $s \equiv 1$, then $z : \mathbf{s} \rightarrow (\tilde{\sigma})^m_*$.

Let $\tilde{\rho} \equiv \tilde{\rho}_1 \times \dots \times \tilde{\rho}_k \rightarrow \tilde{\rho}_0$ and $\ell \equiv \langle \ell_1 \mapsto t_1, \dots, \ell_k \mapsto t_k \rangle$.

Now for any $y_1, \dots, y_n, h_0, h_1, \dots, h_k$ of appropriate types with associates $y'_1, \dots, y'_n, h'_0, h'_1, \dots, h'_k$ with respect to $m_1, \dots, m_n, \ell_1, \dots, \ell_k$ respectively, if $j = 1, \dots, n$ and $i = 0, \dots, k$,

$$Y_j = \begin{cases} y'_j(a), & \text{if } s_j \equiv 0, \\ y'_j, & \text{if } s_j \equiv 1, \end{cases} \quad H_i = \begin{cases} h'_i(a), & \text{if } t_i \equiv 0, \\ h'_i, & \text{if } t_i \equiv 1, \end{cases}$$

$$\begin{aligned}
 g(a, Y_1, \dots, Y_n, H_0, H_1, \dots, H_k) &= (\lambda(z) f'(a, Y_1, \dots, Y_n, z))(H_0, H_1, \dots, H_k) \\
 &= f'(a, Y_1, \dots, Y_n, H_0, H_1, \dots, H_k) \\
 &= f(y_1, \dots, y_n, h_0, h_1, \dots, h_k, a) \\
 &= g(y_1, \dots, y_n, h_0, h_1, \dots, h_k, a).
 \end{aligned}$$

Thus, g' is an associate of g w.r.t. $\langle m_1 \mapsto s_1, \dots, m_n \mapsto s_n, m \mapsto s, \ell \rangle$. \dashv

The following corollary generalizes Corollary 2.2.10.

Corollary 2.4.9. *If f' is an associate of $f : \tilde{\tau}$ with respect to m and for $x_i : \tilde{\sigma}_i$,*

$$g = \lambda(x_1) \dots \lambda(x_n) f,$$

and $\ell \equiv \langle \ell_1 \mapsto t_1, \dots, \ell_n \mapsto t_n, m \rangle$ is a closed locality input index of its type, then the function g' defined for any $a : s$ by

$$g'(a) = \lambda(z_1) \dots \lambda(z_n)(f'(a))$$

is an associate of g with respect to ℓ .

Chapter 3

Locality of Terms

In Chapter 2, we described formally the locality behavior of typed objects. In this chapter, we will use these notions to study the locality character of $\mathsf{L}_{\text{ar}}^\lambda$ terms.

In Section 3.1, we define *local* terms and their relation to local objects. For general terms, the definition of an associate of the denotation of a term $A : \tilde{\sigma}$ with respect to a closed locality input index is not straightforward since one has to keep track of the locality dependencies between the subterms of A . In Section 3.2, this is made precise with the use of *closed locality proofs* which are suitably labeled term formation trees.

In Section 3.3, we formulate and prove the main technical result of the thesis, the existence of a *most local* closed locality proof of a term A which represents the “most local” possible locality behavior of A . We will use this result in Chapter 5 to define the factual content of a term A at any state.

3.1 Local Terms

The denotational semantics of $\mathsf{L}_{\text{ar}}^\lambda$ associate an object $\text{den}(A) \in \mathbb{T}_{\tilde{\sigma}}$ with every closed term $A : \tilde{\sigma}$. Would it be enough to characterize local terms as those terms whose denotation is a local object? Although tempting, the idea is not correct — all terms of type $\tilde{\mathfrak{t}}$ that render natural language sentences would be local under this “definition”. Any account of locality of terms which cannot discriminate between $\text{run}(\text{John}) : \tilde{\mathfrak{t}}$ and $\Box(\text{run}(\text{John})) : \tilde{\mathfrak{t}}$ misses something crucial and cannot be used in any useful way.

At the core of the terms of $\mathsf{L}_{\text{ar}}^\lambda(K)$ that render natural language expressions are the constants in the set K . The locality behaviors of constants are given by their interpretation and in turn, they characterize the behavior of

the terms in which they occur. This is expressed in the following definition and it is justified by the lemma that follows it.

Definition 3.1.1. (Local Term) A term A is *local* if the denotation of any constant that occurs in it is a local object.

By this definition, $\text{run}(\text{John})$ is a local term while $\Box(\text{run}(\text{John}))$ and $\text{former}(\text{minister}, \text{John})$ are not. On the other hand, a term with no constants is always local — a case that is of little interest for our consideration here.

Proposition 3.1.2. *If a term $A : \tilde{\sigma}$ is local and its free variables are in the list $x_1 : \tilde{\sigma}_1, \dots, x_n : \tilde{\sigma}_n$, then the function*

$$f_A(h_1, \dots, h_n) = \text{den}(A)(g\{x_1 := h_1, \dots, x_n := h_n\})$$

is a local object.

Proof. The proof is by induction on the formation rules of the terms of $\mathbf{L}_{\text{ar}}^\lambda$ and it uses the properties of local objects in Theorem 2.2.9.

(1) $A \equiv c$. Then,

$$f_c(h_1, \dots, h_n) = \text{den}(c)(g\{x_1 := h_1, \dots, x_n := h_n\}) = c.$$

Since c is local, by Corollary 2.2.10, f_c is also local.

(2) $A \equiv x : \tilde{\sigma}$, where x is a pure or recursive variable. If $x \equiv x_i$, then

$$f_x(h_1, \dots, h_n) = \text{den}(x)(g\{x_1 := h_1, \dots, x_i := h_i, \dots, x_n := h_n\}) = h_i.$$

By Theorem 2.2.9, projection functions are local, and thus f_x is local.

(3) $A \equiv B(C)$. Suppose that the free variables of both of B and C are in the common list x_1, \dots, x_n . By induction hypothesis, the proposition holds for B and C , that is, f_B and f_C are local, and

$$f_{B(C)}(h_1, \dots, h_n) = f_B(h_1, \dots, h_n, f_C(h_1, \dots, h_n)).$$

By the composition case in Theorem 2.2.9, $f_{B(C)}$ is local.

(4) $A \equiv \lambda(v)(B)$. Suppose that the free variables of B are in the list x_1, \dots, x_n, v . By induction hypothesis, f_B is local, and

$$\begin{aligned} f_{\lambda(v)(B)}(h_1, \dots, h_n) &= \text{den}(\lambda(v)(B))(g\{x_1 := h_1, \dots, x_n := h_n\}) \\ &= \left(h \mapsto \text{den}(B)(g\{x_1 := h_1, \dots, x_i := h_i, \dots, x_n := h_n, v := h\}) \right). \end{aligned}$$

By the λ -abstraction case in Theorem 2.2.9, $f_{\lambda(v)B}$ is local.

(5) $A \equiv A_0$ where $\{p_1 := A_1, \dots, p_n := A_n\}$. Suppose that the free variables of all A_i are in the common list of x_1, \dots, x_m and p_1, \dots, p_n . By induction hypothesis, the proposition holds for A_0, A_1, \dots, A_n , that is, $f_{A_0}, f_{A_1}, \dots, f_{A_n}$ are local. For any appropriately typed objects h_1, \dots, h_m

$$\begin{aligned} f_A(h_1, \dots, h_m) &= \text{den}(A_0)(g\{x_1 := h_1, \dots, x_m := h_m, p_1 := P_1, \dots, p_n := P_n\}) \\ &= f_{A_0}(h_1, \dots, h_m, P_1, \dots, P_n), \end{aligned}$$

where each P_i is defined by recursion on $\text{rank}(p_i)$ as in Section 1.5.1.

By a generalized version of the composition case in Theorem 2.2.9, f_A is local if the objects P_1, \dots, P_n as functions over h_1, \dots, h_m are local. The proof that each P_i is local is by induction on the rank of p_i .

(i) $\text{rank}(p_i) = 0$. Then, for any h_1, \dots, h_m of appropriate types

$$P_i(h_1, \dots, h_m) = f_{A_i}(h_1, \dots, h_m)$$

which is local by induction hypothesis of the proposition.

(ii) $\text{rank}(p_i) = k \geq 1$. By induction hypothesis, all P_{j_1}, \dots, P_{j_r} where the ranks of all p_{j_1}, \dots, p_{j_r} are lower than k , are local. Then, for any h_1, \dots, h_m of appropriate types

$$P_i(h_1, \dots, h_m) = f_{A_i}(h_1, \dots, h_m, P_{j_1}, \dots, P_{j_r}),$$

and thus, P_i is again local by the composition case of Theorem 2.2.9. \dashv

3.2 Closed Locality Proofs

In the case of general terms, we will explore in this section how the associate of the denotation of a term A with respect to a closed locality input index ℓ of its type depends on the associates of its subterms. We have to understand the *locality dependencies* of the locality indices that respect and follow the $\mathbb{L}_{\text{ar}}^\lambda$ -term formation rules.

Example 3.2.1. For example, consider the closed term $\text{former}(\text{president}) : \tilde{\mathbf{e}} \rightarrow \tilde{\mathbf{t}}$ and the closed locality input index $\ell \equiv \langle l \mapsto 0 \rangle$ of its type. If $\ell_1 \equiv \langle \langle l \mapsto t_1 \rangle \mapsto 1, \ell \rangle$ and $\ell_2 \equiv \langle l \mapsto t_3 \rangle$ are closed locality input indices of former and president , respectively, then, if we consider their corresponding preferred associates, by the evaluation case in Theorem 2.4.8, for any $a : \mathbf{s}$

$$(\text{former}(\text{president}))_*^\ell(a) = (\text{former})_*^{\ell_1}(a, (\text{president})_*^{\ell_2}).$$

So, it must be the case that $t_1 \equiv t_3$ and the locality output index of *president* must be equal to 1. The restriction is quite natural since $\langle l \mapsto t_1 \rangle$ is the part of ℓ_1 which describes the way the first argument of *former* treats its own argument and thus it should coincide with ℓ_2 .

Such restrictions imitate in a way type restrictions applied in the formation rules of the terms of $\mathcal{L}_{\text{ar}}^\lambda$. We will use formation trees of terms suitably labeled as tools for coding these restrictions in a way close to our intuitions about them.

For any term A and its formation tree, we describe in what follows a decoration of it (*closed locality proof of A*) such that each subterm is labeled by a closed locality input index of its type. In the case of a constant $c : \tilde{\sigma}$ that occurs in a term A , we are only interested in the closed locality input indices of its type such that its denotation c has an associate with respect to them. We call them the *closed locality input indices of a constant c* .

Definition 3.2.2. (Closed Locality Proof of a Term) A *closed locality proof* (Π_A) of a term A is an annotated version of its formation tree such that each node is decorated by a label of the form

$$A' : \ell \mapsto t$$

where A' is a term, ℓ is a closed locality input index of $\text{type}(A')$ and t is an index constant. We say that A' is *indexed* by $\ell \mapsto t$ in Π_A .

A closed locality proof of a term A is defined by applying the following rules:

(LP-CON) If $A \equiv c$, ℓ is a closed locality input index of c and t is any index constant, then

$$\Pi_c := c : \ell \mapsto t$$

is a closed locality proof of c .

(LP-VAR) If $A \equiv x$ where x is a pure or recursive variable, ℓ is a closed locality input index of $\text{type}(x)$ and t is any index constant, then

$$\Pi_x := x : \ell \mapsto t$$

is a closed locality proof of x .

(LP-APP) If $A \equiv B(C)$, Π_B and Π_C are locality proofs of B and C respectively and t' is an index constant, then

$$\Pi_{B(C)} := \frac{\begin{array}{c} \Pi_B \\ \vdots \\ B : \langle \ell_1 \mapsto t_1, \ell_2 \rangle \mapsto t \end{array} \quad \begin{array}{c} \Pi_C \\ \vdots \\ C : \ell_1 \mapsto t_1 \end{array}}{B(C) : \ell_2 \mapsto t'}$$

is a closed locality proof of $B(C)$ provided that

1. Every pure or recursive variable that occurs free in both B and C is indexed by the same closed locality index in both Π_B and Π_C .
2. If a pure or recursive variable x that occurs free in $B(C)$ is indexed by $\ell_x \mapsto t_x$ in $\Pi_{B(C)}$, then $t' \leq t_x$.

(LP- λ -INTRO) If $A \equiv \lambda(v)(B)$, Π_B is a locality proof of B , $\ell_0 \mapsto t_0$ is a closed locality index of the type of v and t' is an index constant, then

$$\Pi_{\lambda(v)(B)} := \frac{\begin{array}{c} \Pi_B \\ \vdots \\ B : \ell \mapsto t \end{array}}{\lambda(v)(B) : \langle \ell_0 \mapsto t_0, \ell \rangle \mapsto t'}$$

is a closed locality proof of $\lambda(v)(B)$ provided that

1. If v occurs free in B , then it is indexed in Π_B by $\ell_0 \mapsto t_0$.
2. If a pure or recursive variable x that occurs free in $\lambda(v)(B)$ is indexed by $\ell_x \mapsto t_x$ in $\Pi_{\lambda(v)(B)}$, then $t' \leq t_x$.

(LP-REC) If $A \equiv A_0$ where $\{p_1 := A_1, \dots, p_n := A_n\}$, $\Pi_{A_0}, \Pi_{A_1}, \dots, \Pi_{A_n}$ are locality proofs of A_0, A_1, \dots, A_n respectively, then

$$\Pi_A := \frac{\begin{array}{c} \Pi_{A_0} \\ \vdots \\ A_0 : \ell_0 \mapsto t \end{array} \quad \begin{array}{c} \Pi_{A_1} \\ \vdots \\ A_1 : \ell_1 \mapsto t_1 \end{array} \quad \dots \quad \begin{array}{c} \Pi_{A_n} \\ \vdots \\ A_n : \ell_n \mapsto t_n \end{array}}{A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\} : \ell_0 \mapsto t}$$

is a closed locality proof of A_0 where $\{p_1 := A_1, \dots, p_n := A_n\}$ provided that

1. Every pure or recursive variable that occurs free in more than one of the parts A_0, \dots, A_n is indexed by the same closed locality index in all its occurrences in the corresponding locality proofs.

$$\begin{array}{c}
\frac{\text{and} : \langle l \mapsto 0, l \mapsto 0 \rangle \mapsto 0 \quad \frac{c_1 : \langle l \mapsto 1 \rangle \mapsto 0 \quad x : l \mapsto 1}{c_1(x) : l \mapsto 0}}{\text{and}(c_1(x)) : \langle l \mapsto 0 \rangle \mapsto 0} \quad \frac{c_2 : \langle l \mapsto 1 \rangle \mapsto 0 \quad x : l \mapsto 1}{c_2(x) : l \mapsto 0}}{\frac{\text{and}(c_1(x), c_2(x)) : l \mapsto 0}{\lambda(x)\text{and}(c_1(x), c_2(x)) : \langle l \mapsto 1 \rangle \mapsto 0}}
\end{array}$$

Figure 3.1: A closed locality proof of $\lambda(x)\text{and}(c_1(x), c_2(x))$.

-
2. For each $i = 1, \dots, n$, p_i is indexed by $\ell_i \mapsto t_i$ in any Π_{A_j} in which it occurs free.
 3. If a pure or recursive variable x that occurs free in A is indexed by $\ell_x \mapsto t_x$ in Π_A , then $t \leq t_x$.

If Π_A is a closed locality proof of A , then the node labeled by $A : \ell \mapsto t$ is the *root* of the proof and $A : \ell \mapsto t$ is its *root label*. If $A' : \ell' \mapsto t'$ is the label of a node in Π_A , the closed locality proof with root label $A' : \ell' \mapsto t'$ is the *subproof* of A' defined by Π_A (Notation: $\Pi_A^{A'}$).

It is useful to make some remarks and give some examples of locality proofs in order to elucidate the definition and motivate the constraints that are imposed by it.

First, unlike all the other rules where the output index of the root label is arbitrary, in the (LP-REC) rule, the output index of the recursive term is equal to the output index of its head. Now, the output index of a term codifies the local or non local way that it will be used. Suppose that the output index of a recursive term is equal to 1 while the output index of its head is equal to 0. Then, although the computation of the denotation of a recursive term is defined by the computation of the denotation of its head, the second is used locally while the first is used non locally.

Second, by a simple inspection on the rules and especially, the constraints imposed by (LP-APP) and (LP-REC) according to which any free occurrence of a variable in the corresponding subproofs is indexed by the same closed locality index, it follows that:

Proposition 3.2.3. *In a closed locality proof of A , all the free occurrences of a pure or recursive variable x are indexed by the same closed locality index.*

Example 3.2.4. Let $A \equiv \lambda(x)\text{and}(c_1(x), c_2(x))$ be a term where $\text{and} : \tilde{\mathbf{t}} \times \tilde{\mathbf{t}} \rightarrow \tilde{\mathbf{t}}$ and both c_1 and c_2 are constants of type $\tilde{\mathbf{e}} \rightarrow \tilde{\mathbf{t}}$. Suppose that c_1 has only

$$\frac{\frac{\text{love} : \langle l \mapsto 0, l \mapsto 0 \rangle \mapsto 0 \quad p : l \mapsto 0}{\text{love}(p) : \langle l \mapsto 0 \rangle \mapsto 0} \quad p : l \mapsto 0}{\frac{\text{love}(p, p) : l \mapsto 0 \quad \text{John} : l \mapsto 0}{\text{love}(p, p) \text{ where } \{p := \text{John}\} : l \mapsto 0}}$$

Figure 3.2: A closed locality proof of $\text{love}(p, p)$ where $\{p := \text{John}\}$.

one closed locality input index, $\langle l \mapsto 1 \rangle$, while c_2 has both $\langle l \mapsto 0 \rangle$ and $\langle l \mapsto 1 \rangle$ and that the constant `and` is local. In Figure 3.1, a closed locality proof of A is shown that respects the given closed locality input indices of the constants.

Notice that since c_1 is indexed by $\langle l \mapsto 1 \rangle$, all the free occurrences of x in the subterm $\text{and}(c_1(x), c_2(x))$ must be indexed by $l \mapsto 1$ and thus, we must also use the input index $\langle l \mapsto 1 \rangle$ for c_2 . But why x must be indexed by the same index in both its occurrences? The answer is quite simple — if x is a pure variable, in order to apply the (LP- λ -INTRO) in a term A in which x occurs, there must be a unique index for it in all its occurrences in A . This index determines the index of the first argument of the new term $\lambda(x)(A)$ and it surely depends on how x itself behaves in terms of locality within A .

Suppose now that p is a recursive variable. Then, analogously, p must be indexed by the same index in order to apply the (LP-REC) rule which bounds its occurrences.

Example 3.2.5. Consider the term $\text{love}(p, p)$ where $\{p := \text{John}\}$ and its closed locality proof that is shown in Figure 3.2. By the corresponding constraint of the (LP-REC) rule, the part of the recursive term `John` must be indexed by the same closed locality index as p since in the recursive term we have the assignment $p := \text{John}$. Thus, by the time that we are to apply the (LP-REC) rule, p must be indexed by the same index in all its occurrences — in this example, by $l \mapsto 0$.

Unlike variables and their indices that are described in Proposition 3.2.3, if a constant c occurs in two different places in a term A , then it may be indexed by two different closed locality input indices of its type as long as they are both such that its denotation c has an associate with respect to them.

Example 3.2.6. An example is shown in Figure 3.3 of a term that, using co-ordination as it is presented in [22], renders the sentence ‘The temperature

$$\begin{array}{c}
\text{the} : \langle \langle l \mapsto 0 \rangle \mapsto 1 \rangle \mapsto 0 \quad \text{temp} : \langle l \mapsto 0 \rangle \mapsto 1 \\
\hline
\text{and} : \langle l \mapsto 0, l \mapsto 0 \rangle \mapsto 0 \quad \text{the(temp)} : l \mapsto 1 \quad \text{the} : \langle \langle l \mapsto 0 \rangle \mapsto 0 \rangle \mapsto 0 \quad \text{humid} : \langle l \mapsto 0 \rangle \mapsto 0 \\
\hline
\text{and}(x(\text{the(temp)})) : \langle l \mapsto 0 \rangle \mapsto 0 \quad x : \langle l \mapsto 1 \rangle \mapsto 0 \quad \text{the(humid)} : l \mapsto 1 \\
\hline
\text{and}(x(\text{the(temp)}), x(\text{the(humid)})) : l \mapsto 0 \\
\hline
\lambda(x)\text{and}(x(\text{the(temp)}), x(\text{the(humid)})) : \langle \langle l \mapsto 1 \rangle \mapsto 0 \rangle \mapsto 0 \quad \text{rise} : \langle l \mapsto 1 \rangle \mapsto 0 \\
\hline
\lambda(x)(\text{and}(x(\text{the(temp)}), x(\text{the(humid)})))(\text{rise}) : l \mapsto 0
\end{array}$$

Figure 3.3: A closed locality proof of the term $\lambda(x)(\text{and}(x(\text{the(temp)}), x(\text{the(humid)})))(\text{rise})$.

and the humidity are rising'¹. In this proof, the constant `the` is indexed by two different locality input indices.

Now, notice that in all the three rules, (LP-APP), (LP- λ -INTRO) and (LP-REC), there is a common constraint:

Proposition 3.2.7. *All the free occurrences of any pure or recursive variable x in a closed locality proof of A with root label $A : \ell \mapsto 1$ have locality output index 1.*

Proof. It follows directly by a simple inspection on the rules. \dashv

We explain the importance of this constraint in the case of (LP-APP) rule in the following example.

Example 3.2.8. Consider the closed locality subproof of the term $\text{rise}(\text{the}(x))$ shown in Figure 3.4. Since *rise* uses its first argument non locally, the only closed locality input index of the constant *rise* is $\langle l \mapsto 1 \rangle$. The application of (LP-APP) rule in (*) forces the locality output index of $\text{the}(x)$ to be 1. Now, x occurs free in the subproof of $\text{the}(x)$ and by the constraint of the (LP-APP) rule, its locality output index must also be equal to 1.

The idea is simple — x is within the scope of a non local subterm of A (in this case *rise*) and this means that x itself cannot be marked as used locally. If we are to quantify over it by applying the λ -abstractor, the new term must use its first argument non locally. Thus, the term $\lambda(x)\text{rise}(\text{the}(x))$ has locality input index $\langle \langle l \mapsto 0 \rangle \mapsto 1 \rangle$.

¹We assume that the constant $\text{temp} : \tilde{e} \rightarrow \tilde{t}$ renders the word ‘temperature’ and the constant $\text{humid} : \tilde{e} \rightarrow \tilde{t}$ ‘humidity’.

$$\begin{array}{c}
(*) \frac{\text{rise} : \langle l \mapsto 1 \rangle \mapsto 0 \quad \frac{\text{the} : \langle \langle l \mapsto 0 \rangle \mapsto 1 \rangle \mapsto 0 \quad x : \langle l \mapsto 0 \rangle \mapsto 1}{\text{the}(x) : l \mapsto 1}}{\text{rise}(\text{the}(x)) : l \mapsto 0}} \\
\hline
\lambda(x)\text{rise}(\text{the}(x)) : \langle \langle l \mapsto 0 \rangle \mapsto 1 \rangle \mapsto 0
\end{array}$$

Figure 3.4: A closed locality proof of $\lambda(x)\text{rise}(\text{the}(x))$.

The full technical role of the constraints imposed by all the rules will be clarified further in Chapter 4 where closed locality proofs will be used in order to define formally the associates of the denotation of a term A .

Suppose now Π_A is a closed locality proof of a term A with root label $A : \ell \mapsto t$. The question that follows is simple: is there an associate of the denotation function of A with respect to ℓ ? By the definition of a closed locality proof of a term A , if a constant c occurs in A , it is labeled in any proof of A by an input index with respect to which its denotation has an associate. Similarly thus to the case of the local terms, the answer is positive and this is proved in the following theorem which generalizes Proposition 3.1.2.

Theorem 3.2.9. (Associate of the Denotation of a Term with respect to a Closed Locality Proof) *Let Π_A be a closed locality proof of a term A with root label*

$$A : \ell \mapsto t.$$

If the free variables of A are in the list $x_1 : \tilde{\sigma}_1, \dots, x_n : \tilde{\sigma}_n$ and for $i = 1, \dots, n$, x_i is indexed in Π_A by $\ell_i \mapsto t_i$, then the function f defined by

$$f(h_1, \dots, h_n) = \text{den}(A)(g\{x_1 := h_1, \dots, x_n := h_n\})$$

has an associate with respect to the closed locality input index

$$\langle \ell_1 \mapsto t_1, \dots, \ell_n \mapsto t_n, \ell \rangle.$$

Proof. The proof is by induction on A and it uses the properties of associates shown in Theorem 2.4.8.

(1) $A \equiv c$. Then, if $\Pi_c \equiv c : \ell \mapsto t$,

$$f_c(h_1, \dots, h_n) = \text{den}(c)(g\{x_1 := h_1, \dots, x_n := h_n\}) = c$$

and by hypothesis and Corollary 2.4.9, it has an associate with respect to $\langle \ell_1 \mapsto t_1, \dots, \ell_n \mapsto t_n, \ell \rangle$ where for $i = 1, \dots, n$, ℓ_i is a closed locality input index of type(x_i).

(2) $A \equiv x : \tilde{\sigma}$ where x is a pure or recursive variable and for some i , $x_i \equiv x$. Suppose $\Pi_x \equiv x : \ell \mapsto t$, and

$$f_x(h_1, \dots, h_n) = \text{den}(x)(g\{x_1 := h_1, \dots, x_i := h_i, \dots, x_n := h_n\}) = h_i.$$

By case (i) of Theorem 2.4.8, if, for $i = 1, \dots, n$, ℓ_i is a closed locality input index of $\text{type}(x_i)$, f_x has an associate w.r.t. $\langle \ell_1 \mapsto t_1, \dots, \ell_n \mapsto t_n, \ell_i \rangle$.

(3) $A \equiv B(C)$. Suppose that both B and C have free variables in the common list x_1, \dots, x_n . Let $\Pi_{B(C)}$ be a closed locality proof of $B(C)$ with root label $B(C) : \ell \mapsto t'$ and suppose that for $i = 1, \dots, n$, each x_i is indexed in it by $\ell_i \mapsto t_i$.

$$\Pi_{B(C)} : \frac{\begin{array}{c} \Pi_B \\ \vdots \\ B : \langle \ell_0 \mapsto t_0, \ell \rangle \mapsto t \end{array} \quad \begin{array}{c} \Pi_C \\ \vdots \\ C : \ell_0 \mapsto t_0 \end{array}}{B(C) : \ell \mapsto t'}$$

By induction hypothesis, $\langle \ell_1 \mapsto t_1, \dots, \ell_n \mapsto t_n, \ell_0 \mapsto t_0, \ell \rangle$ is a closed locality input index of f_B , $\langle \ell_1 \mapsto t_1, \dots, \ell_n \mapsto t_n, \ell_0 \rangle$ is an index of f_C , and

$$f_{B(C)}(h_1, \dots, h_n) = f_B(h_1, \dots, h_n, f_C(h_1, \dots, h_n)).$$

By Proposition 3.2.7, the hypothesis of the composition case in Theorem 2.4.8 are all met and thus, it follows that $f_{B(C)}$ has an associate with respect to $\langle \ell_1 \mapsto t_1, \dots, \ell_n \mapsto t_n, \ell \rangle$.

(4) $A \equiv \lambda(v)(B)$. Let $\Pi_{\lambda(v)(B)}$ be a closed locality proof of $\lambda(v)(B)$ and suppose that for $i = 1, \dots, n$, each x_i is indexed in it by $\ell_i \mapsto t_i$ and if v occurs free in Π_B , then it is indexed by $\ell_0 \mapsto t_0$.

$$\Pi_{\lambda(v)(B)} \equiv \frac{\begin{array}{c} \Pi_B \\ \vdots \\ B : \ell \mapsto t \end{array}}{\lambda(v)(B) : \langle \ell_0 \mapsto t_0, \ell \rangle \mapsto t'}$$

By induction hypothesis, $\langle \ell_1 \mapsto t_1, \dots, \ell_n \mapsto t_n, \ell_0 \mapsto t_0, \ell \rangle$ is a closed locality input index of f_B , and

$$\begin{aligned} f_{\lambda(v)(B)}(h_1, \dots, h_n) &= \text{den}(\lambda(v)(B))(g\{x_1 := h_1, \dots, x_n := h_n\}) \\ &= (h \mapsto f_B(h_1, \dots, h_n, h)). \end{aligned}$$

Thus, by λ -abstraction case in Theorem 2.4.8, $f_{\lambda(v)(B)}$ has an associate with respect to $\langle \ell_1 \mapsto t_1, \dots, \ell_n \mapsto t_n, \ell_0 \mapsto t_0, \ell \rangle$.

(5) $A \equiv A_0$ where $\{p_1 := A_1, \dots, p_n := A_n\}$. Suppose that the free variables of all A_i are in list $x_1, \dots, x_m, p_1, \dots, p_n$ and let i, k range over $1, \dots, n$ and j over $1, \dots, m$. Let Π_A be a closed locality proof of A .

$$\Pi_A := \frac{\begin{array}{ccc} \Pi_{A_0} & \Pi_{A_1} & \Pi_{A_n} \\ \vdots & \vdots & \vdots \end{array}}{A_0 : \ell_0 \Downarrow t \quad A_1 : \ell_1 \Downarrow t_1 \dots A_n : \ell_n \Downarrow t_n} \quad A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\} : \ell_0 \Downarrow t$$

For any appropriately typed h_1, \dots, h_m

$$f_A(h_1, \dots, h_m) = f_{A_0}(h_1, \dots, h_m, P_1, \dots, P_n)$$

where each P_i is defined by recursion on $\text{rank}(p_i)$ as a function h_1, \dots, h_m as in Section 1.5.1.

If each x_j is indexed by $\ell_{x_j} \Downarrow t_{x_j}$, then by induction hypothesis, f_{A_0} has an associate with respect to $\langle \ell_{x_j} \Downarrow t_{x_j}, \ell_i \Downarrow t_i, \ell_0 \rangle$. By a generalized version of the composition case in Theorem 2.4.8, it is enough to show that each P_i has an associate with respect to $\langle \ell_{x_j} \Downarrow t_{x_j}, \ell_k \Downarrow t_k, \ell_i \rangle$ where $\text{rank}(p_k) < \text{rank}(p_i)$. The proof is by induction on the $\text{rank}(p_i)$.

(i) $\text{rank}(p_i) = 0$. It follows directly by the induction hypothesis of this theorem.

(ii) $\text{rank}(p_i) \geq 1$. Suppose that p_{r_1}, \dots, p_{r_k} have rank lower than $\text{rank}(p_i)$. Then, by induction hypothesis, P_{r_1}, \dots, P_{r_k} are such that they have associates with the appropriate locality indices. Then, for any h_1, \dots, h_m

$$P_i(h_1, \dots, h_m) = f_{A_i}(h_1, \dots, h_m, P_{r_1}, \dots, P_{r_k}).$$

Since f_{A_i} has an associate w.r.t. $\langle \ell_{x_j} \Downarrow t_{x_j}, \ell_{r_1} \Downarrow t_{r_1}, \dots, \ell_{r_k} \Downarrow t_{r_k}, \ell_i \rangle$, again by the composition case of Theorem 2.4.8 and by induction hypothesis for P_{r_1}, \dots, P_{r_k} , P_i has an associate with respect to $\langle \ell_{x_j} \Downarrow t_{x_j}, \ell_i \rangle$.

Notice that Proposition 3.2.7 again guarantees that we can indeed apply the composition case of Theorem 2.4.8 in all cases. \dashv

By this theorem, given closed locality input indices ℓ_c for every constant c that occurs in a term A , if we can construct a closed locality proof of A such that the constants are indexed by the given indices, then the denotation function of A has an associate with respect to the closed locality input index of the root label. In other words, the theorem describes a criterion based on which the locality behavior of a term A is determined by the locality behavior of the constants that occur in it.

$$\begin{array}{c}
\text{former} : \langle \langle l \mapsto 0 \rangle \mapsto 1, l \mapsto 0 \rangle \mapsto 0 \quad \text{minister} : \langle l \mapsto 0 \rangle \mapsto 1 \\
\hline
\text{former}(\text{minister}) : \langle l \mapsto 0 \rangle \mapsto 0 \quad \text{John} : l \mapsto 0 \\
\hline
\text{former}(\text{minister}, \text{John}) : l \mapsto 0
\end{array}$$

Figure 3.5: A closed locality proof of $\text{former}(\text{minister}, \text{John})$.

Example 3.2.10. Consider the term $A \equiv \text{former}(\text{minister}, \text{John}) : \tilde{\tau}$. The fact that there is an associate with respect to l does not give any information about the locality behavior of the term. Suppose now that the constants of the term are indexed as follows: $\text{former} : \langle \langle l \mapsto 0 \rangle \mapsto 1, l \mapsto 0 \rangle$, $\text{minister} : \langle l \mapsto 0 \rangle$ and $\text{John} : l$. In Figure 3.5, it is shown that there is a closed locality proof of A with respect to them.

Furthermore, by Theorem 2.4.8, we know exactly how an associate of the object $\text{den}(\text{former}(\text{minister}, \text{John}))(g)$ is computed with respect to the associates of the denotations of its subterms. If we consider the corresponding preferred associates, for any assignment g and any $a : s$,

$$\begin{aligned}
& (\text{den}(\text{former}(\text{minister}, \text{John}))(g))_*^l(a) \\
&= \text{former}_*^{\langle \langle l \mapsto 0 \rangle \mapsto 1, l \mapsto 0 \rangle}(a, \text{minister}_*^{\langle l \mapsto 0 \rangle}, \text{John}_*^l(a)).
\end{aligned}$$

Finally, we end this section by a direct consequence of the definition of the closed locality proof and Proposition 2.4.6.

Corollary 3.2.11. *For any term A , there is a closed locality proof (called standard) such that all the subterms are indexed by the standard locality index of their type.*

3.3 The Most Local Closed Locality Proof of a Term

For each term A there may be many closed locality proofs that may differ on one or some of the indices that label their nodes.

Example 3.3.1. Consider again the term $\text{former}(\text{minister}, \text{John})$ (Example 3.2.10) and its closed locality proof shown in Figure 3.5. This proof is not unique — an alternative closed locality proof of this term is shown in Figure 3.6. Both the constants former and minister are indexed in this proof with different locality input indices than those in the proof in Figure 3.5.

$$\frac{\frac{\text{former} : \langle \langle l \mapsto 1 \rangle \mapsto 1, l \mapsto 0 \rangle \mapsto 0 \quad \text{minister} : \langle l \mapsto 1 \rangle \mapsto 1}{\text{former}(\text{minister}) : \langle l \mapsto 0 \rangle \mapsto 0} \quad \text{John} : l \mapsto 0}{\text{former}(\text{minister}, \text{John}) : l \mapsto 0}$$

Figure 3.6: An alternative closed locality proof of $\text{former}(\text{minister}, \text{John})$.

Although the locality input index of the root label is still the same, the computation of the associate of the denotation of the term with respect to the associates of the subterms is different.

Finally, notice that even given particular closed locality input indices of the constants that occur in a term A , if there is a proof of A with respect to them, it may be the case that there are more than one.

Now, it is natural to ask if we can compare two arbitrary closed locality proofs of a term A and whether there is a way to express all the closed locality proofs of a term A by some appropriate labeling of its formation tree. The answer to the first question is the definition of a partial order between closed locality proofs of a term. It is based on the natural ordering between the two index constants 0 and 1, namely $0 < 1$, and the corresponding partial order between two locality indices of the same type defined as follows.

Definition 3.3.2. For any two closed locality indices ℓ and ℓ' of the same type $\tilde{\sigma}$, ℓ is *less than or equal to* ℓ' ($\ell \leq \ell'$) (or ℓ is *more local than* ℓ') if:

- (i) If $\tilde{\sigma} \equiv \tilde{\sigma}_0$, $\ell \mapsto t \leq \ell' \mapsto t'$ if and only if $t \leq t'$.
- (ii) If $\tilde{\sigma} \equiv \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}_0$,

$$\langle \ell_1 \mapsto t_1, \ell_2 \rangle \mapsto t \leq \langle \ell'_1 \mapsto t'_1, \ell'_2 \rangle \mapsto t'$$

if and only if $\ell_1 \mapsto t_1 \leq \ell'_1 \mapsto t'_1$, $\ell_2 \leq \ell'_2$ and $t \leq t'$.

For example, if $\tilde{\sigma} \equiv \tilde{\mathbf{e}} \times \tilde{\mathbf{e}} \rightarrow \tilde{\mathbf{t}}$, $\ell_1 \equiv \langle l \mapsto 0, l \mapsto 0 \rangle$ and $\ell_2 \equiv \langle l \mapsto 1, l \mapsto 0 \rangle$, then $\ell_1 \leq \ell_2$. The idea is that for any object f of type $\tilde{\sigma}$, if f is indexed by ℓ_1 then the locality behavior that it expresses is “more local” than that expressed by ℓ_2 . The function f under ℓ_1 treats more arguments locally (or in other cases, its arguments treat their own arguments more locally etc.) than the same function does under ℓ_2 .

Notice that the ordering defined above between locality indices of the same type is partial — if, for example, $\ell_1 \equiv \langle l \mapsto 0, l \mapsto 1 \rangle$ and $\ell_2 \equiv \langle l \mapsto 1, l \mapsto 0 \rangle$, then neither $\ell_1 \leq \ell_2$ nor $\ell_2 \leq \ell_1$.

$$\begin{array}{c}
\frac{y : \langle l \mapsto 0, l \mapsto 1 \rangle \mapsto 0 \quad \frac{\text{the} : \langle \langle l \mapsto 0 \rangle \mapsto 0 \rangle \mapsto 0 \quad x : \langle l \mapsto 0 \rangle \mapsto 0}{\text{the}(x) : l \mapsto 0}}{y(\text{the}(x)) : \langle l \mapsto 1 \rangle \mapsto 0} \\
\\
\frac{y : \langle l \mapsto 1, l \mapsto 0 \rangle \mapsto 0 \quad \frac{\text{the} : \langle \langle l \mapsto 0 \rangle \mapsto 0 \rangle \mapsto 1 \quad x : \langle l \mapsto 0 \rangle \mapsto 0}{\text{the}(x) : l \mapsto 1}}{y(\text{the}(x)) : \langle l \mapsto 0 \rangle \mapsto 0}
\end{array}$$

Figure 3.7: Two closed locality proofs of $y(\text{the}(x))$.

The partial order between indices is generalized to a partial order between two closed locality proofs Π_1 and Π_2 of a term A : $\Pi_1 \leq \Pi_2$ if and only if at each node, if a node of Π_1 is $A' : \ell_1 \mapsto t_1$ and at Π_2 , the corresponding node is $A' : \ell_2 \mapsto t_2$, then $\ell_1 \mapsto t_1 \leq \ell_2 \mapsto t_2$. The formal definition is by recursion:

Definition 3.3.3. For any two closed locality proofs Π_1 and Π_2 of a term A , Π_1 is *more local than* Π_2 ($\Pi_1 \leq \Pi_2$) if and only if

- (i) If $A \equiv c$, $\Pi_1 \equiv c : \ell \mapsto t$ and $\Pi_2 \equiv c : \ell' \mapsto t'$, then $\ell \mapsto t \leq \ell' \mapsto t'$.
- (ii) If $A \equiv x$, $\Pi_1 \equiv x : \ell \mapsto t$ and $\Pi_2 \equiv x : \ell' \mapsto t'$, then $\ell \mapsto t \leq \ell' \mapsto t'$.
- (iii) If $A \equiv B(C)$, the root label of Π_1 is $B(C) : \ell \mapsto t$ and the root label of Π_2 is $B(C) : \ell' \mapsto t'$, then $\Pi_1^B \leq \Pi_2^B$, $\Pi_1^C \leq \Pi_2^C$ and $\ell \mapsto t \leq \ell' \mapsto t'$.
- (iv) If $A \equiv \lambda(v)(B)$, the root label of Π_1 is $\lambda(v)(B) : \ell \mapsto t$ and the root label of Π_2 is $\lambda(v)(B) : \ell' \mapsto t'$, then $\Pi_1^B \leq \Pi_2^B$ and $\ell \mapsto t \leq \ell' \mapsto t'$.
- (v) If $A \equiv A_0$ where $\{p_1 := A_1, \dots, p_n := A_n\}$, the root label of Π_1 is $A : \ell \mapsto t$ and the root label of Π_2 is $A : \ell' \mapsto t'$, then for $i = 0, \dots, n$, $\Pi_1^{A_i} \leq \Pi_2^{A_i}$ and $\ell \mapsto t \leq \ell' \mapsto t'$.

Thus, if for two closed locality proofs Π_1 and Π_2 of a term A , $\Pi_1 \leq \Pi_2$, then Π_1 expresses a “more local” locality behavior of A than Π_2 . We have already seen that the locality behavior of A as it is expressed by a closed locality proof depends on the locality input indices that label the constants that occur in it. Thus, we already know that if a constant c has two locality input indices that are incomparable with respect to the relation ‘less than or equal to’, then the corresponding closed locality proofs of A are incomparable with respect to the ‘more local’ ordering.

Now, suppose that we consider only the closed locality proofs of a term A that are formed with respect to particular closed locality input indices of the

constants that occur in it (or even with respect to comparable indices). It is still not true that we can always order these proofs with respect to the ‘less than or equal to’ relation (see, for example, the two closed locality proofs in Figure 3.7).

Nevertheless, we will show that under certain natural conditions, we can determine a closed locality proof of a term A which is the most local one, that is, it is more local than all its closed locality proofs. To do that, we first try to answer the other question we formulated in page 67 — that is, is there a labeling of the formation tree of A that “expresses” all the possible closed locality proofs of A ?

Definition 3.3.4. (Locality Proof Schema of a Term) A *locality proof schema* of A is a pair

$$(T_A, \text{COND}_A)$$

such that

1. T_A is the formation tree of A annotated by locality indices of the type of each subterm. Their index tokens are all index variables and no index variable occurs more than once.
2. COND_A is a (finite) set of equalities and inequalities of the form

$$b_i = b_j, \quad b_i = 1, \quad b_i \leq b_j$$

where b_i, b_j are index variables that occur in T_A , and

3. For any evaluation $\rho : \{\text{Index Variables}\} \rightarrow \{0, 1\}$, if ρ satisfies the equalities and inequalities of COND_A , then the labeled tree $\rho(T_A)$ is a closed locality proof of A .

Notice that $\rho(T_A)$ is the tree that is formed if we apply ρ on all the index variables that occur in T_A .

Example 3.3.5. Consider a locality proof schema of the term $\Box(\text{run}(x))$ (Figure 3.8). The locality indices that label the subterms of the term have only index variables. The conditions are such that they express the basic constraints that are imposed by the corresponding rules of the closed locality proofs that we use.

Thus, the application of the (LP-APP) rule in (*) forces the conditions $\{b_1 = b_3, b_4 \leq b_3\}$ while in (**) the application of the same rule forces $\{b_4 = b_5, b_7 \leq b_3\}$. The equality $\{b_5 = 1\}$ is forced by the (LP-CON) rule for the constant \Box . In the case of run no equality is needed because both $\langle l \mapsto 0 \rangle$ and

$$\begin{array}{c}
T : \frac{\frac{\square : \langle l \mapsto b_5 \rangle \mapsto b_6 \quad \frac{\text{run} : \langle l \mapsto b_1 \rangle \mapsto b_2 \quad x : l \mapsto b_3}{\text{run}(x) : l \mapsto b_4} (*)}{\square(\text{run}(x)) : l \mapsto b_7} (**)} \\
\text{COND} = \{b_1 = b_3, b_5 = 1, b_4 = b_5, b_7 \leq b_3, b_4 \leq b_3\}
\end{array}$$

Figure 3.8: A locality proof schema of the term $\square(\text{run}(x))$.

$\langle l \mapsto 1 \rangle$ are closed locality input indices of it. It is thus straightforward that for any evaluation $\rho : \{b_1, b_2, b_3, b_4, b_5, b_6, b_7\} \rightarrow \{0, 1\}$ such that ρ satisfies COND, the tree $\rho(T)$ is a closed locality proof of $\square(\text{run}(x))$.

In general, the set of conditions COND_A that are described in the definition of a locality proof schema of a term A are always satisfiable by at least the evaluation which assigns the value 1 to all the variables that occur in the corresponding tree T_A . It is trivial that such an evaluation satisfies any such set of conditions and the corresponding tree is simply the standard closed locality proof of the term A . It is also straightforward that for any A there is at least one locality proof schema — the schema such that there is an equality $\{b_i = 1\}$ in the set of conditions COND for every index variable b_i that occurs in T .

In general, a term A does not have a unique locality proof schema. The tree T_A is of course unique up to alphabetic renaming of the index variables that occur in it but the set of conditions may vary.

Example 3.3.6. Consider, for example, a constant $c : \tilde{\mathbf{e}} \times \tilde{\mathbf{e}} \rightarrow \tilde{\mathbf{t}}$ that has three closed locality input indices: $\langle l \mapsto 0, l \mapsto 1 \rangle$, $\langle l \mapsto 1, l \mapsto 0 \rangle$ and of course, $\langle l \mapsto 1, l \mapsto 1 \rangle$. In Figure 3.9, two of the possible locality proof schemata of c are shown.

To answer our initial question, we need to select a schema (T_A, COND_A) of A such that for any Π_A , ρ is an evaluation such that $\rho(T_A) = \Pi_A$ if and only if ρ satisfies COND_A .

Definition 3.3.7. (Generic Locality Proof Schema of a Term) A *generic locality proof schema* of A is a locality proof schema (T_A, COND_A) of A such that every closed locality proof of A is equal to $\rho(T_A)$ for some evaluation ρ that satisfies COND_A .

First, we use this definition in the following trivial proposition.

$$c : \langle l \mapsto b_1, l \mapsto b_2 \rangle \mapsto b_3$$

$$\text{COND}_c = \{b_1 = 1\} \quad \text{or} \quad \text{COND}'_c = \{b_2 = 1\}$$

Figure 3.9: Locality proof schemata of c .

Proposition 3.3.8. *Suppose that (T_A, COND_A) is a generic locality proof schema of A . Then, a closed locality proof Π_1 of A is more local than Π_2 ($\Pi_1 \leq \Pi_2$) if and only if, given evaluations ρ_1 and ρ_2 such that they satisfy COND_A and $\rho_1(T_A) = \Pi_1$ and $\rho_2(T_A) = \Pi_2$, for any index variable b that occurs in T_A , $\rho_1(b) \leq \rho_2(b)$.*

Secondly, it is not always true that a term A has a generic locality proof schema (see for example the constant c in Example 3.3.6). Theorem 3.3.11 describes under what circumstances a term A has a generic schema but in what follows, we show first that any term A has at most one.

Definition 3.3.9. Two set of equalities and inequalities, as in Definition 3.3.4, COND_1 and COND_2 over the same (up to alphabetic renaming) set of index variables are *equivalent* if, for every evaluation ρ , ρ satisfies COND_1 if and only if an alphabetic variant of ρ satisfies COND_2 .

For example, the two sets $\{b_i = b_j, b_i = 1\}$ and $\{b'_j = 1, b'_j \leq b'_i\}$ are equivalent.

Proposition 3.3.10. *Every term $A : \tilde{\sigma}$ has at most one (up to alphabetic renaming of the index variables and to equivalent set of conditions) generic locality proof schema.*

Proof. Suppose that a term A has two generic locality proof schemata: (T_A, COND_A) and (T'_A, COND'_A) . Then, as we have already mentioned in the definition of a locality proof schema, T_A and T'_A are identical up to alphabetic renaming of the index variables that occur in them.

Now, COND_A and COND'_A may include different equalities and inequalities, involving even different index variables. We will show though that they are equivalent.

Suppose ρ is an evaluation over the index variables of T_A such that ρ satisfies COND_A . Thus, $\rho(T_A) = \Pi_A$ where Π_A is a closed locality proof of A . By definition, there is an evaluation ρ' over the index variables of T'_A such that it satisfies COND'_A and $\rho'(T'_A) = \Pi_A$. \dashv

We have already pointed out that the locality behavior of a term A depends on the locality behaviors of the constants that occur in it. In order thus to understand when a term has a generic locality proof schema, we should start by investigating whether any state-dependent typed constant in K has one. We have already seen in Example 3.3.6 that this is not in general the case. In other words, we can always add a new constant in K that does not have a generic schema and thus, refute any such claim.

On the other hand, we should notice that a closed input index ℓ of a constant $c : \tilde{\sigma}$ is not attributed to it arbitrarily. It is a closed locality input index of its denotation c — that is, by Chapter 2, c has an associate with respect to it. Thus, in the case of the constant in Example 3.3.6, we should instead provide its denotation function and then, prove that it has an associate with respect to the three given locality input indices.

In general, the interpretation of the constants of K of the $L_{ar}^\lambda(K)$ is coupled with their locality behavior. It is the denotation of $\text{rise} : \tilde{\mathbf{e}} \rightarrow \tilde{\mathbf{t}}$ that determines the non local treatment of its argument which is “coded” in its input index $\langle l \mapsto 1 \rangle$. The generic locality proof schema of constant rise is

$$T_{\text{rise}} \equiv \text{rise} : \langle l \mapsto b_1 \rangle \mapsto b_2, \quad \text{COND}_{\text{rise}} = \{b_1 = 1\}.$$

The non local treatment of an argument of the denotation of a constant, or in other words, its “intensional character”, is a locality feature that characterizes it and in a generic schema, it is expressed by the conditions of the form $b_i = 1$. The local treatment of an argument of an object, in contrast, does not “force” the non local one. Thus, the generic locality proof schema of former is

$$T_{\text{former}} \equiv \text{former} : \langle \langle l \mapsto b_1 \rangle \mapsto b_2, l \mapsto b_3 \rangle \mapsto b_4, \quad \text{COND}_{\text{former}} = \{b_2 = 1\}.$$

Consider now an example of a constant whose generic schema needs the use of inequalities between index variables. Let $\text{eval} : (\tilde{\mathbf{e}} \rightarrow \tilde{\mathbf{t}}) \times \tilde{\mathbf{e}} \rightarrow \tilde{\mathbf{t}}$ be such that for $f_1 : \tilde{\mathbf{e}} \rightarrow \tilde{\mathbf{t}}, f_2 : \tilde{\mathbf{e}}, a : \mathbf{s}$,

$$\text{eval}(f_1, f_2, a) = 1 \iff f_1(f_2, a) = 1.$$

For example, consider the denotation of the terms $\text{eval}(\text{rise}, \text{the}(\text{temp}))$ and $\text{eval}(\text{run}, \text{John})$. The generic locality proof schema of this constant is

$$T_{\text{eval}} \equiv \text{eval} : \langle \langle l \mapsto b_1 \rangle \mapsto b_2, l \mapsto b_3 \rangle \mapsto b_4, \quad \text{COND}_{\text{eval}} = \{b_1 \leq b_3\}.$$

It is an open issue whether the constants in K that render natural language expressions are such that their locality behavior can be fully described with

the use of conditions introduced here. It is though clear that this can be done at least for all the ones that come from rather natural examples.

Finally, notice also that the equalities and inequalities between index variables that are included in the conditions of a generic locality proof schema are used also in expressing the constraints of the locality proofs.

Theorem 3.3.11. *Suppose that every constant that occurs in a term A has a generic locality proof schema. Then,*

- (i) *A has a generic locality proof schema, and*
- (ii) *A has a most local closed locality proof.*

Proof. (i) The proof is by induction on A .

(1) $A \equiv c$. It holds by hypothesis.

(2) $A \equiv x$. Trivially, the generic locality proof schema of x is simply a pair (T_x, COND_x) where $T_x \equiv x : \ell \mapsto t$, where $\ell \mapsto t$ is the generic locality index of the type of x , and $\text{COND}_x = \emptyset$.

(3) $A \equiv B(C)$. Suppose that (T_B, COND_B) and (T_C, COND_C) are the generic locality proofs schemata of B and C , respectively with distinct index variables. If $\ell'_2 \mapsto b'$ is the generic locality index of $B(C)$ with fresh index variables, then

$$T_{B(C)} := \frac{\begin{array}{cc} T_B & T_C \\ \vdots & \vdots \\ B : \langle \ell_1 \mapsto b_1, \ell_2 \rangle \mapsto b & C : \ell'_1 \mapsto b'_1 \end{array}}{B(C) : \ell'_2 \mapsto b'}$$

and the set of conditions is

$$\begin{aligned} \text{COND}_{B(C)} &:= \text{COND}_B \cup \text{COND}_C \\ &\cup \{ \ell_1 \mapsto b_1 = \ell'_1 \mapsto b'_1 \} \cup \{ \ell_2 = \ell'_2 \} \\ &\cup \{ \ell_x \mapsto b_x = \ell'_x \mapsto b'_x \mid \text{for any } x \text{ such that } x : \ell_x \mapsto b_x \text{ occurs} \\ &\quad \text{free in } T_B \text{ and } x : \ell'_x \mapsto b'_x \text{ in } T_C \} \\ &\cup \{ b' \leq b_x \mid \text{for any } x \text{ that occurs free in } T_{B(C)} \text{ with output} \\ &\quad \text{index } b_x \}. \end{aligned}$$

Notice that each time, the constraint imposed by the rule is expressed by a set of conditions that are included in $\text{COND}_{B(C)}$. For example, in order to have $\ell_2 = \ell'_2$, we add a set of equalities between all the corresponding pairs of index variables that occur in ℓ_2 and ℓ'_2 .

The pair $(T_{B(C)}, \text{COND}_{B(C)})$ is a locality proof schema of $B(C)$ since the equalities and inequalities that are added in $\text{COND}_{B(C)}$ reflect the constraints of the (LP-APP) rule and thus, every evaluation ρ that satisfies them is such that $\rho(T_{B(C)})$ is a closed locality proof of $B(C)$.

We show next that this schema is generic. Suppose a closed locality proof $\Pi_{B(C)}$ of $B(C)$ and let Π_B and Π_C be the subproofs of B and C respectively that are defined by it.

$$\Pi_{B(C)} \equiv \frac{\begin{array}{cc} \Pi_B & \Pi_C \\ \vdots & \vdots \\ B : \langle \ell \mapsto t_1, \ell' \rangle \mapsto t & C : \ell \mapsto t_1 \end{array}}{B(C) : \ell' \mapsto t'}$$

By induction hypothesis, there are evaluations ρ_B and ρ_C that satisfy COND_B and COND_C , respectively such that $\rho_B(T_B) = \Pi_B$ and $\rho_C(T_C) = \Pi_C$.

Consider now the substitution ρ_0 over the index variables of the root label of $T_{B(C)}$ such that

$$\rho_0(\ell'_2 \mapsto b') = \ell' \mapsto t'.$$

The claim is that the substitution

$$\rho = \rho_B \cup \rho_C \cup \rho_0$$

is such that $\rho(T_{B(C)}) = \Pi_{B(C)}$ and ρ satisfies $\text{COND}_{B(C)}$. It follows immediately if we consider that

- Since T_B , T_C and $\ell'_2 \mapsto b'$ have distinct index variables, ρ_B , ρ_C and ρ_0 act on disjoint set of index variables.
- Since $\Pi_{B(C)}$ is a closed locality proof of $B(C)$, ρ_B and ρ_C satisfy $\text{COND}_{B(C)}$, that is,
 - $\rho_B(\ell_1 \mapsto b_1) = \rho_C(\rho'_1 \mapsto b'_1) = \ell \mapsto t_1$,
 - if $x : \ell_x \mapsto b_x$ occurs in T_B and $x : \ell'_x \mapsto b'_x$ occurs in T_C , then $\rho_B(\ell_x \mapsto b_x) = \rho_C(\ell'_x \mapsto b'_x)$ and finally,
 - if $x : \ell_x \mapsto b_x$ occurs free in $T_{B(C)}$, then, if, for example, it occurs free in B , $\rho_0(b') \leq \rho_B(b_x)$.
- ρ_0 acts only on fresh variables and it simply adds the equalities that are imposed by the $\rho_0(\ell'_2 \mapsto b') = \ell' \mapsto t'$, thus, it respects $\text{COND}_{B(C)}$ by definition.

(4) $A \equiv \lambda(v)(B)$. Suppose that (T_B, COND_B) is a generic locality proof schema of B . If $\langle \ell_v \mapsto b_v, \ell' \rangle \mapsto b'$ is the generic locality index of the type of $\lambda(v)(B)$ with fresh index variables, then

$$T_{\lambda(v)(B)} := \frac{\begin{array}{c} T_B \\ \vdots \\ B : \ell \mapsto b \end{array}}{\lambda(v)(B) : \langle \ell_v \mapsto b_v, \ell' \rangle \mapsto b'}$$

and the set of conditions are

$$\begin{aligned} \text{COND}_{\lambda(v)(B)} &:= \text{COND}_B \cup \{\ell = \ell'\} \\ &\cup \{\ell_v \mapsto b_v = \ell'_v \mapsto b'_v \mid \text{if } v : \ell'_v \mapsto b'_v \text{ occurs free in } T_B\} \\ &\cup \{b' \leq b_x \mid \text{for any } x \text{ that occurs free in } T_{\lambda(v)(B)} \text{ with output} \\ &\quad \text{index } b_x\}. \end{aligned}$$

The equalities and inequalities that are added in $\text{COND}_{\lambda(v)(B)}$ reflect the constraints of the (LP- λ -INTRO) rule and thus, every evaluation ρ that satisfies them is such that $\rho(T_{\lambda(v)(B)})$ is a closed locality proof of $\lambda(v)(B)$.

Suppose $\Pi_{\lambda(v)(B)}$ is a closed locality proof of $\lambda(v)(B)$ and Π_B the closed locality subproof of B that is defined by it.

$$\Pi_{\lambda(v)(B)} \equiv \frac{\begin{array}{c} \Pi_B \\ \vdots \\ B : \ell_1 \mapsto t_1 \end{array}}{\lambda(v)(B) : \langle \ell_0 \mapsto t_0, \ell_1 \rangle \mapsto t}$$

If ρ_B is an evaluation such that $\rho_B(T_B) = \Pi_B$ and ρ_0 is such that

$$\rho_0(\langle \ell_v \mapsto b_v, \ell' \rangle \mapsto b') = \langle \ell_0 \mapsto t_0, \ell_1 \rangle \mapsto t,$$

it follows easily that

$$(\rho_B \cup \rho_0)(T_{\lambda(v)(B)}) = \Pi_{\lambda(v)(B)}.$$

(5) $A \equiv A_0$ where $\{p_1 := A_1, \dots, p_n := A_n\}$. Suppose that for $i = 0, \dots, n$, $(T_{A_i}, \text{COND}_{A_i})$ are generic locality proof schemata of A_i with distinct index variables. If $\ell'_0 \mapsto b'$ is a generic locality index of the type of A with fresh index variables, then

$$T_A := \frac{\begin{array}{ccc} T_{A_0} & T_{A_1} & T_{A_n} \\ \vdots & \vdots & \vdots \end{array}}{A_0 : \ell_0 \mapsto b \quad A_1 : \ell_1 \mapsto b_1 \quad \dots \quad A_n : \ell_n \mapsto b_n \quad \text{where } \{p_1 := A_1, \dots, p_n := A_n\} : \ell'_0 \mapsto b'}$$

and the set of conditions, analogously, are

$$\begin{aligned} \text{COND}_A &\equiv \text{COND}_{A_0} \cup \dots \cup \text{COND}_{A_n} \cup \{\ell_0 \mapsto b = \ell'_0 \mapsto b'\} \\ &\cup \{\ell_x \mapsto b_x = \ell'_x \mapsto b'_x \mid \text{for any } x \text{ such that } x : \ell_x \mapsto b_x \text{ occurs} \\ &\quad \text{free in some } T_{A_i} \text{ and } x : \ell'_x \mapsto b'_x \text{ in } T_{A_j}\} \\ &\cup \{\ell_{p_i} \mapsto b_{p_i} = \ell_i \mapsto b_i \mid \text{if } p_i : \ell_{p_i} \mapsto b_{p_i} \text{ occurs free in some } A_j\} \\ &\cup \{b' \leq b_x \mid \text{for any } x \text{ occurring free in } T_A \text{ with output} \\ &\quad \text{index } b_x\}. \end{aligned}$$

As in the previous cases, it is trivial that any evaluation ρ that satisfies COND_A is such that $\rho(T_A)$ is a closed locality proof of A and thus, the pair (T_A, COND_A) is locality proof schema of A .

Suppose now that Π_A is a closed locality proof of A and $\Pi_{A_0}, \dots, \Pi_{A_n}$ are the closed locality proofs of A_0, \dots, A_n respectively, that it defines.

$$\Pi_A \equiv \frac{\begin{array}{ccc} \Pi_{A_0} & \Pi_{A_1} & \Pi_{A_n} \\ \vdots & \vdots & \vdots \end{array}}{A_0 : \ell \mapsto t_0 \quad A_1 : \ell'_1 \mapsto t_1 \quad \dots \quad A_n : \ell'_n \mapsto t_n \quad \text{where } \{p_1 := A_1, \dots, p_n := A_n\} : \ell \mapsto t_0}$$

If, for $i = 0, \dots, n$, $\rho_{A_i}(T_{A_i}) = \Pi_{A_i}$ and ρ_0 is such that

$$\rho_0(\ell'_0 \mapsto b') = \ell \mapsto t_0,$$

it follows analogously that

$$(\rho_{A_0} \cup \dots \cup \rho_{A_n} \cup \rho_0)(T_A) = \Pi_A.$$

(ii) We want to prove that there is a closed locality proof of A , Π_0 , such that for every closed locality proof Π_A of A , $\Pi_0 \leq \Pi_A$.

Let (T_A, COND_A) be a generic locality proof schema of A , let V be the finite set of index variables that occur in T_A and let W be, initially, the empty set.

We will describe a process of removing index variables from V and putting them into W based on whether they must be assigned the value 1 or not.

Repeatedly, we do the following:

1. For every $b \in V$, if $\{b = 1\} \subseteq \text{COND}_A$, we update the sets as follows: $V = V \setminus \{b\}$ and $W = W \cup \{b\}$.
2. For every $b \in V$, if $\{b = b'\} \subseteq \text{COND}_A$ or $\{b' \leq b\} \subseteq \text{COND}_A$, where $b' \in W$, the sets are updated as follows: $V = V \setminus \{b\}$ and $W = W \cup \{b\}$.

Notice that this process terminates since the set V is finite.

Now, trivially, any evaluation ρ that satisfies COND_A must be such that for any $b \in W$, $\rho(b) = 1$. Consider now the evaluation ρ_0 such that

$$\rho_0(b) = \begin{cases} 1, & \text{if } b \in W \\ 0, & \text{if } b \in V. \end{cases}$$

First, ρ_0 satisfies COND_A . It is enough to notice that for any $b \in V$, there is no equality of the form $b = 1$, $b = b'$ or $b' \leq b$ where $b' \in W$, and thus, trivially, if we assign the value 0 to all of them the possible equalities and inequalities are satisfied.

Second, for any ρ such that it satisfies COND_A , for any b , $\rho_0(b) \leq \rho(b)$. Thus, if Π_0 is such that $\rho_0(T_A) = \Pi_0$ and Π is such that $\rho(T_A) = \Pi$, then $\Pi_0 \leq \Pi$. \dashv

This is the main technical result of the thesis in that for each term A , the most local closed locality proof of A represents the locality behavior which respects the locality behavior of the constants that occur in A and keeps track of the locality dependencies in such a way that the computations involved are as local as possible.

Chapter 4

Formal Associates of Terms

In Chapter 3, we defined the closed locality proofs of each term $A : \tilde{\sigma}$ so that if the root label is $A : \ell \Downarrow t$, then there is an associate of the denotation of A with respect to ℓ . The locality proofs of a term A keep track of the locality dependencies of its subterms and characterize its locality behavior. In this chapter, for each term A , each closed locality proof Π_A of A with root label $A : \ell \Downarrow t$ and a state variable u , we define, through an appropriate syntactical transformation, the *associate term* $A_{\Pi_A}^{*,u}$ which defines formally an associate of the denotation of A with respect to ℓ^1 .

In Section 4.1, we present an extension of $\mathsf{L}_{\text{ar}}^\lambda, \mathsf{L}_{\text{ar}}^{\lambda*}$, into which the new associate terms will be defined. We then define the *local associate transformation* for local terms (Section 4.2) and the corresponding *associate transformation* for general terms (Section 4.3). In Chapter 5, we will use the associate term of any term A at a state a with respect to the most local locality proof of its canonical form to define the *factual content* of A at a .

4.1 An Extension of $\mathsf{L}_{\text{ar}}^\lambda$

As in the case of LIL with the intension and extension operators and in the case of $\mathsf{L}_{\text{ar}}^\lambda$ with the recursive construct, we will enrich $\mathsf{L}_{\text{ar}}^\lambda$ in order to define the associates of terms formally in this extended language, $\mathsf{L}_{\text{ar}}^{\lambda*}$.

For each constant $c : \tilde{\sigma} \in K$ and each closed locality input index ℓ of c , we introduce in $\mathsf{L}_{\text{ar}}^\lambda(K)$ a new constant $c_\ell^* : s \rightarrow (\tilde{\sigma})_*^\ell$ which denotes the

¹For short, an associate of the denotation of A with respect to the index ℓ will be simply called an associate of the term A with respect to ℓ .

preferred associate of c with respect to ℓ , i.e.,

$$\text{den}(\mathbf{c}_\ell^*)(g) = c_*^\ell = (\text{den}(\mathbf{c})(g))_*^\ell.$$

For each type $\tilde{\sigma}$ and each closed locality input index ℓ of $\tilde{\sigma}$, we also introduce, for convenience, infinitely many new pure and recursive variables,

$$x_\ell^*, p_\ell^* : \mathbf{s} \rightarrow (\tilde{\sigma})_*^\ell.$$

Finally, we introduce a new formation rule, the *associate application* which behaves exactly like the ordinary functional application but is reserved for the newly introduced constants and variables. Thus, the terms of the extended language, $\mathbf{L}_{\text{ar}}^{\lambda*}$, are defined by the recursion:

$$\begin{aligned} A : \equiv & \mathbf{c} \mid x \mid \mathbf{c}_\ell^*[u] \mid x_\ell^*[u] \mid B(C) \mid \lambda(v)(B) \\ & \mid A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\} \end{aligned} \quad (4.1)$$

where x is a pure or recursive variable, u and v are pure variables and p_1, \dots, p_n are recursive variables.

Types are assigned as for the terms in (1.8) in Section 1.5.1 and since associate application behaves like ordinary application,

$$\begin{aligned} & \text{if } \mathbf{c}_\ell^* : \mathbf{s} \rightarrow (\tilde{\sigma})_*^\ell \text{ and } u : \mathbf{s}, \text{ then } \mathbf{c}_\ell^*[u] : (\tilde{\sigma})_*^\ell, \\ & \text{if } x_\ell^* : \mathbf{s} \rightarrow (\tilde{\sigma})_*^\ell \text{ and } u : \mathbf{s}, \text{ then } x_\ell^*[u] : (\tilde{\sigma})_*^\ell. \end{aligned}$$

Notice that u occurs free in $\mathbf{c}_\ell^*[u]$ and both u and x_ℓ^* occur free in $x_\ell^*[u]$.

In what concerns the denotational semantics of $\mathbf{L}_{\text{ar}}^{\lambda*}$, its interpretation structure is again \mathcal{U} — the interpretation structure of $\mathbf{L}_{\text{ar}}^\lambda$ defined in Section 1.5.1. The denotation function defined in this structure for the newly introduced terms of associate application is given by

$$\begin{aligned} \text{den}(\mathbf{c}_\ell^*[u])(g) &= c_*^\ell(g(u)) \\ \text{den}(x_\ell^*[u])(g) &= g(x_\ell^*)(g(u)). \end{aligned}$$

Before defining the canonical form of the terms of the extended language, we have to reconsider the immediate terms in $\mathbf{L}_{\text{ar}}^{\lambda*}$. Consider the following terms defined by associate application for a pure v and a recursive variable p where ℓ is a locality input index of the appropriate type:

$$v_\ell^*[u] \text{ and } p_\ell^*[u].$$

They are treated as *generalized variables* (*pure* and *recursive*, respectively) which means that the *immediate* terms of $\mathcal{L}_{\text{ar}}^{\lambda*}$ are

$$X ::= V_i \mid P(V_1, \dots, V_n) \mid \lambda(v_1, \dots, v_m, u)(P(V_1, \dots, V_n)) \quad (4.2)$$

$$V_i ::= v_i \mid v_{i,\ell_i}^*[u_i], \quad P ::= p \mid p_\ell^*[u_i]$$

where $v_i, v_{i,\ell_i}^*, v_1, \dots, v_m, u_i, u$ are pure variables and p and p_ℓ^* are recursive variables. For example, the following terms are immediate:

$$p_\ell^*[u](v_{\ell'}^*[u]), \quad p_\ell^*(u, v_{\ell'}^*[u]), \quad \lambda(u)p_\ell^*(u, v_{\ell'}^*[u]).$$

Notice that if $v : \tilde{\sigma} \rightarrow \tilde{\tau}$ and $w : \tilde{\sigma}$ are pure variables, then, whereas the term $v(w)$ is not immediate in $\mathcal{L}_{\text{ar}}^\lambda$, the term $v_\ell^*[u]$ where $u : \mathbf{s}$ is immediate in $\mathcal{L}_{\text{ar}}^{\lambda*}$. The idea is that associate application is going to be used in particular cases to compute the value of an object at a state, and thus, as in the case of applications like $p(w)$ where p is a recursive variable, should be performed at no “computational cost” in the Reduction Calculus of the extended $\mathcal{L}_{\text{ar}}^{\lambda*}$.

The Reduction Calculus in $\mathcal{L}_{\text{ar}}^{\lambda*}$ is exactly the same as before (Table 1.3), and thus, the associate applications terms $c_\ell^*[u]$ and $x_\ell^*[u]$ are irreducible.

Proposition 4.1.1. (Theorem 3.12 in [22])

- (a) *Constants and immediate terms are irreducible.*
- (b) *Associate application terms $c_\ell^*[u]$ and $x_\ell^*[u]$ are irreducible.*
- (c) *An application term $B(C)$ is irreducible if and only if C is immediate and B is explicit and irreducible.*
- (d) *A λ -term $\lambda(u)(B)$ is irreducible if and only if B is explicit and irreducible.*
- (e) *A recursive term A_0 where $\{p_1 := A_1, \dots, p_n := A_n\}$ is irreducible if and only if all the parts A_0, \dots, A_n are irreducible.*

Proof. Case (b) is straightforward by inspecting the reduction rules. \dashv

Notice also that the congruence relation is the same as before (page 19) and since associate application terms are irreducible, we have

$$\begin{aligned} \text{cf}(c_\ell^*[u]) &\equiv c_\ell^*[u] \text{ where } \{ \} \\ \text{cf}(x_\ell^*[u]) &\equiv x_\ell^*[u] \text{ where } \{ \}. \end{aligned}$$

The Canonical Form Theorem (Theorem 1.5.1) extends here with no change, which means that in $\mathcal{L}_{\text{ar}}^{\lambda*}$, for any term A there is unique (up to congruence), irreducible recursive term $\text{cf}(A)$, such that $A \Rightarrow \text{cf}(A)$.

4.2 Formal Local Associates of Local Terms

In this section, we define the *local associate transformation* which associates a term $A^{*,u} : \sigma$ in $\mathbb{L}_{\text{ar}}^{\lambda*}$ with any local term $A : \tilde{\sigma}$ in $\mathbb{L}_{\text{ar}}^{\lambda}$ and any state variable $u : \mathbf{s}$ which does not occur free in A ,

$$A : \tilde{\sigma}, u : \mathbf{s} \mapsto A^{*,u} : \sigma.$$

This new term defines within $\mathbb{L}_{\text{ar}}^{\lambda*}$ a local associate of the original term, that is, if we set for any state $a : \mathbf{s}$,

$$A^{*,\bar{a}} \equiv A^{*,u} \{u := \bar{a}\},$$

then,

$$(a \mapsto \text{den}(A^{*,\bar{a}})(g)) \text{ is a local associate of } \text{den}(A)(g).$$

Before defining the local associate transformation, we will make a simplification on the notation of the previous section. For each constant $\mathbf{c} : \tilde{\sigma} \in K$, if ℓ is its local closed locality input index, then we simply omit it, that is, we let

$$\mathbf{c}^* \equiv \mathbf{c}_{\ell}^*.$$

Analogously, if ℓ is the local closed locality input index of the type of a variable $x : \tilde{\sigma}$, we let

$$x^* \equiv x_{\ell}^*.$$

We will also say that an occurrence of a subterm $x^*[u]$ (or similarly, $x_{\ell}^*[u]$) where x is a pure or recursive variable *is free in a term A* if the occurrences of both variables x^* (or x_{ℓ}^*) and u are free in A .

Proposition 4.2.1. (Local Associate Transformation) *For any local term $A : \tilde{\sigma}$ and any fresh state variable $u : \mathbf{s}$, there is a term $A^{*,u} : \sigma$ in $\mathbb{L}_{\text{ar}}^{\lambda*}$, such that:*

- (i) *If A has free variables in the list $x_1 : \tilde{\sigma}_1, \dots, x_n : \tilde{\sigma}_n$, then $A^{*,u} : \sigma$ has free variables in the list $x_1^* : \sigma_1, \dots, x_n^* : \sigma_n, u : \mathbf{s}$.*
- (ii) *For any variable x , x occurs free in A if and only if the subterm $x^*[u]$ occurs free in $A^{*,u}$.*
- (iii) *The following conditions hold:*

- (1) $\mathbf{c}^{*,u} \equiv \mathbf{c}^*[u]$
- (2) $x^{*,u} \equiv x^*[u]$
- (3) $(B(C))^{*,u} \equiv B^{*,u}(C^{*,u})$

- (4) $(\lambda(v)(B))^{*,u} \equiv \lambda(v')(B^{*,u}\{v^*[u] := v'\})$, where v' is a fresh pure variable of the type of $v^*[u]$
- (5) $(A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\})^{*,u}$
 $\equiv (A_0)^{*,u}\{p_i^*[u] := q_i \mid 1 \leq i \leq n\} \text{ where}$
 $\{q_1 := (A_1)^{*,u}\{p_i^*[u] := q_i \mid 1 \leq i \leq n\},$
 \vdots
 $q_n := (A_n)^{*,u}\{p_i^*[u] := q_i \mid 1 \leq i \leq n\}\}.$

where for $i = 1, \dots, n$, q_i is a fresh recursive variable of the type of $p_i^*[u]$.

Proof. The proof is by a straightforward induction on A .

Notice that the substitutions in each case are free since each time the variables v' and q_i are fresh.

If A is a recursive term, by (ii), for $k = 0, \dots, n$, some p_i occurs (free) in A_k if and only if q_i occurs (free) in $(A_k)^{*,u}$. It follows that for $i = 1, \dots, n$, $\text{rank}(p_i) = \text{rank}(q_i)$ and thus, $A^{*,u}$ is a recursive term. \dashv

For example, if $q, x' : e$ are fresh variables,

$$\begin{aligned} (\lambda(x)\text{love}(x, x))^{*,u} &\equiv \lambda(x')\text{love}^*[u](x', x') \\ (\text{run}(p) \text{ where } \{p := \text{John}\})^{*,u} &\equiv \text{run}^*[u](q) \text{ where } \{q := \text{John}^*[u]\}. \end{aligned}$$

Theorem 4.2.2. *If $A : \tilde{\sigma}$ with free variables in the list $x_1 : \tilde{\sigma}_1, \dots, x_n : \tilde{\sigma}_n$ is a local term, the function $f_A : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}$ is defined by*

$$f_A(f_1, \dots, f_n) = \text{den}(A)(g\{x_1 := f_1, \dots, x_n := f_n\})$$

and the function $f_{A^{*,u}} : s \times \sigma_1 \times \dots \times \sigma_n \rightarrow \sigma$ is defined by

$$\begin{aligned} f_{A^{*,u}}(a, h_1, \dots, h_n) \\ = \text{den}(A^{*,u}\{x_1^*[u] := x'_1, \dots, x_n^*[u] := x'_n\}) \\ (g\{x'_1 := h_1, \dots, x'_n := h_n, u := a\}), \end{aligned}$$

where $x'_1 : \sigma_1, \dots, x'_n : \sigma_n$ are fresh variables, then the function $f_{A^{*,u}}$ is a local associate of f_A .

In particular, if A is closed, then for any state $a : s$, the function $(a \mapsto \text{den}(A^{*,\bar{a}}))$ is a local associate of $\text{den}(A)$.

Proof. The proof is by induction on A . It will be useful to use the following abbreviations.

$$\begin{aligned} g\{x_i := f_i\} &\equiv g\{x_1 := f_1, \dots, x_n := f_n\} \\ g\{x'_i := h_i\} &\equiv g\{x'_1 := h_1, \dots, x'_n := h_n\} \end{aligned}$$

(1) $A \equiv c$. Simply, $f_c(f_1, \dots, f_n) = c$, and

$$f_{c^*,u}(a, h_1, \dots, h_n) = \text{den}(c^*[u])(g\{x'_i := h_i, u := a\}) = c^*(a).$$

Thus, by Corollary 2.2.10, $f_{c^*,u}$ is a local associate of f_c .

(2) $A \equiv x$. Suppose that for some j , $x_j \equiv x$.

$f_x(f_1, \dots, f_n) = \text{den}(x)(g\{x_i := f_i\}) = f_j$, and

$$\begin{aligned} f_{x^*,u}(a, h_1, \dots, h_n) \\ = \text{den}(x^*[u]\{x_j^*[u] \equiv x'_j\})(g\{x'_i := h_i, u := a\}) = h_j. \end{aligned}$$

By case (i) of Theorem 2.2.9, $f_{x^*,u}$ is a local associate of f_x .

(3) $A \equiv B(C)$. Suppose that the free variables of B and C are in the list $x_1 : \tilde{\sigma}_1, \dots, x_n : \tilde{\sigma}_n$.

$$\begin{aligned} f_{B(C)}(f_1, \dots, f_n) &= \text{den}(B)(g\{x_i := f_i\})(\text{den}(C)(g\{x_i := f_i\})) \\ &= f_B(f_1, \dots, f_n, f_C(f_1, \dots, f_n)). \end{aligned}$$

$$\begin{aligned} f_{(B(C))^*,u}(a, h_1, \dots, h_n) \\ = \text{den}(B^{*,u}\{x_1^*[u] \equiv x'_1, \dots, x_n^*[u] \equiv x'_n\})(g\{x'_i := h_i, u := a\}) \\ \left(\text{den}(C^{*,u}\{x_1^*[u] \equiv x'_1, \dots, x_n^*[u] \equiv x'_n\})(g\{x'_i := h_i, u := a\}) \right) \\ = f_{B^*,u}(a, h_1, \dots, h_n, f_{C^*,u}(a, h_1, \dots, h_n)). \end{aligned}$$

By induction hypothesis, $f_{B^*,u}$ and $f_{C^*,u}$ are local associates of f_B and f_C respectively, and thus, by the composition case of Theorem 2.2.9, the function $f_{(B(C))^*,u}$ is a local associate of $f_{B(C)}$.

(4) $A \equiv \lambda(v)(B)$. Suppose that the free variables of B are in the list of $x_1 : \tilde{\sigma}_1, \dots, x_n : \tilde{\sigma}_n, v : \tilde{\sigma}$.

$$\begin{aligned} f_{\lambda(v)(B)}(f_1, \dots, f_n) &= \left(f \mapsto \text{den}(B)(g\{x_i := f_i, v := f\}) \right) \\ &= \left(f \mapsto f_B(f_1, \dots, f_n, f) \right). \end{aligned}$$

$$\begin{aligned}
& f_{(\lambda(v)(B))^*, u}(a, h_1, \dots, h_n) \\
&= \text{den}(\lambda(v')(B^{*, u}\{v^*[u] := v'\})\{x_1^*[u] := x'_1, \dots, x_n^*[u] := x'_n\}) \\
&\quad (g\{x'_i := h_i, u := a\}) \\
&= (h \mapsto \text{den}(B^{*, u}\{v^*[u] := v', x_1^*[u] := x'_1, \dots, x_n^*[u] := x'_n\}) \\
&\quad (g\{x'_i := h_i, u := a, v' := h\})) \\
&= (h \mapsto f_{B^{*, u}}(a, h_1, \dots, h_n, h)).
\end{aligned}$$

By induction hypothesis, $f_{B^{*, u}}$ is a local associate of f_B , and thus, by the λ -abstraction case of Theorem 2.2.9, the function $f_{(\lambda(v)(B))^*, u}$ is a local associate of $f_{\lambda(v)(B)}$.

(5) $A \equiv A_0$ where $\{p_1 := A_1, \dots, p_m := A_m\}$. Suppose that the free variables of A are in the list x_1, \dots, x_n and thus, the free variables of each A_i ($i = 1, \dots, m$) are in the list $x_1, \dots, x_n, p_1, \dots, p_m$.

$$\begin{aligned}
f_A(f_1, \dots, f_n) &= \text{den}(A)(g\{x_i := f_i\}) \\
&= \text{den}(A_0)(g\{x_i := f_i, p_1 := P_1, \dots, p_m := P_m\}) \\
&= f_{A_0}(f_1, \dots, f_n, P_1, \dots, P_m),
\end{aligned}$$

where for $i = 1, \dots, m$, each

$$P_i = f_{P_i}(f_1, \dots, f_n)$$

is defined recursively on the $\text{rank}(p_i)$:

$$P_i = f_{A_i}(f_1, \dots, f_n, P_{r_1}, \dots, P_{r_k})$$

where p_{r_1}, \dots, p_{r_k} are the variables with $\text{rank} < \text{rank}(p_i)$ (see also Section 1.5.1).

By recursion on $\text{rank}(q_i)$ again, let

$$Q_i = f_{Q_i}(a, h_1, \dots, h_n) = f_{(A_i)^*, u}(a, h_1, \dots, h_n, Q_{r_1}, \dots, Q_{r_k}).$$

It is important to notice here that by Proposition 4.2.1, for each $i = 1, \dots, m$, $\text{rank}(p_i) = \text{rank}(q_i)$.

Lemma. For $r = 1, \dots, n$, f_{Q_i} is a local associate of f_{P_i} .

Proof. The proof is by induction on the $\text{rank}(p_i)$.

(i) $\text{rank}(p_i) = 0$. It follows trivially since $f_{P_i} = f_{A_i}$ and $f_{Q_i} = f_{(A_i)^*, u}$.

(ii) $\text{rank}(p_i) \geq 1$. Suppose that for all the variables p_{r_1}, \dots, p_{r_k} such that $\text{rank}(p_{r_j}) < \text{rank}(p_i)$, the induction hypothesis hold. It follows easily by the composition case of Theorem 2.2.9, since

$$f_{P_i}(f_1, \dots, f_n) = f_{A_i}(f_1, \dots, f_n, f_{P_{r_1}}(f_1, \dots, f_n), \dots, f_{P_{r_k}}(f_1, \dots, f_n))$$

and

$$\begin{aligned} & f_{Q_i}(a, h_1, \dots, h_n) \\ &= f_{(A_i)^*, u}(a, h_1, \dots, h_n, f_{Q_{r_1}}(a, h_1, \dots, h_n), \dots, f_{Q_{r_k}}(a, h_1, \dots, h_n)). \quad \dashv \end{aligned}$$

Now, by induction hypothesis, the function $f_{(A_0)^*, u}$ is a local associate of f_{A_0} and

$$\begin{aligned} & f_{A^*, u}(a, h_1, \dots, h_n) \\ &= \text{den}(A^{*, u} \{x_1^*[u] \equiv x'_1, \dots, x_n^*[u] \equiv x'_n\})(g\{x'_i := h_i, u := a\}) \\ &= \text{den}((A_0)^{*, u} \{p_1^*[u] \equiv q_1, \dots, p_m^*[u] \equiv q_m, x_1^*[u] \equiv x'_1, \dots, x_n^*[u] \equiv x'_n\}) \\ &\quad (g\{x'_i := h_i, q_1 := Q_1, \dots, q_m := Q_m, u := a\})) \\ &= f_{(A_0)^*, u}(a, h_1, \dots, h_n, Q_1, \dots, Q_m). \end{aligned}$$

Thus, by the composition case of Theorem 2.2.9, $f_{A^*, u}$ is a local associate of f_A . \dashv

The formal local associate of the canonical form of local term $A : \tilde{\sigma}$ is used in Chapter 5 to define the factual content of A at a state a . Finally, we end this section by a simple lemma that we will use in that definition.

Lemma 4.2.3. *If $A : \tilde{\sigma}$ is a local irreducible term in $\mathbb{L}_{\text{ar}}^\lambda$, then the term $A^{*, u} : \sigma$ is irreducible in $\mathbb{L}_{\text{ar}}^{\lambda^*}$.*

Proof. The proof is by a straightforward induction on A , using the syntactical definition of irreducible terms (Proposition 4.1.1). As part of the proof, we show that if $X : \tilde{\sigma}$ is a local immediate term in $\mathbb{L}_{\text{ar}}^\lambda$, then for any state variable $u : \mathbf{s}$, $X^{*, u}$ is immediate in $\mathbb{L}_{\text{ar}}^{\lambda^*}$, which is again straightforward by the following.

- (1) If $X \equiv x$ where x is a pure or recursive variable, then $x^{*, u} \equiv x^*[u]$.
- (2) If $X \equiv p(v_1, \dots, v_n)$, then

$$(p(v_1, \dots, v_n))^{*, u} \equiv p^*[u](v_1^*[u], \dots, v_n^*[u]).$$

- (3) If $X \equiv \lambda(v_1, \dots, v_n)p$, then $(\lambda(v_1, \dots, v_n)p)^{*, u} \equiv \lambda(z_1, \dots, z_n)p^*[u]$, where z_1, \dots, z_n are fresh pure variables with appropriate types.

(4) If $X \equiv \lambda(w_1, \dots, w_m)p(v_1, \dots, v_n)$, then

$$\begin{aligned} & (\lambda(w_1, \dots, w_m)p(v_1, \dots, v_n))^{*,u} \\ & \equiv \lambda(z_1, \dots, z_m)(p^*[u](v_1^*[u], \dots, v_n^*[u])\{w_1^*[u] \equiv z_1, \dots, w_m^*[u] \equiv z_m\}), \end{aligned}$$

where z_1, \dots, z_m are fresh pure variables of appropriate types. \dashv

4.3 Formal Associates of General Terms

Suppose $A : \tilde{\sigma}$ is a term such that its denotation has an associate with respect to a closed locality input index ℓ . In Chapter 3, we showed that syntactically this is witnessed by an existence of, at least, one closed locality proof with root label $A : \ell \Downarrow t$. In general, a term A has multiple closed locality proofs with the same root locality input index. Each one of them describes a different instance of locality behaviors of the subterms of A that, nevertheless, converge to the same closed locality input index of A .

Locality proofs will be proven indispensable in the definition of the corresponding *associate transformation* for general terms since, as it was shown in the case of local terms, it is a syntactical transformation that depends on the locality dependencies of the subterms of a term. Thus, given a closed locality proof Π_A of a term $A : \tilde{\sigma}$ with root label $A : \ell \Downarrow t$, we associate with A and any state variable $u : \mathbf{s}$ that does not occur free in A , an *associate term* $A_{\Pi_A}^{*,u}$, that is,

$$A : \tilde{\sigma}, \Pi_A \text{ with root label } A : \ell \Downarrow t, u : \mathbf{s} \mapsto A_{\Pi_A}^{*,u} : \begin{cases} (\tilde{\sigma})_*^\ell, & \text{if } t \equiv 0 \\ \mathbf{s} \rightarrow (\tilde{\sigma})_*^\ell, & \text{if } t \equiv 1 \end{cases}$$

which defines (roughly) an associate of A with respect to ℓ .

Proposition 4.3.1. (Associate Transformation) *For any term $A : \tilde{\sigma}$ in $\mathbb{L}_{\text{ar}}^\lambda$ and a closed locality proof Π_A with root label $A : \ell \Downarrow t$ and any state variable $u : \mathbf{s}$ that does not occur free in A , there is a term*

$$A_{\Pi_A}^{*,u} : \begin{cases} (\tilde{\sigma})_*^\ell, & \text{if } t \equiv 0 \\ \mathbf{s} \rightarrow (\tilde{\sigma})_*^\ell, & \text{if } t \equiv 1 \end{cases}$$

such that:

(i) If A has free variables in the list $x_1 : \tilde{\sigma}_1, \dots, x_n : \tilde{\sigma}_n$ and for $i = 1, \dots, n$, x_i is indexed by $m_i \Downarrow s_i$ in Π_A , if $t \equiv 1$, then $A_{\Pi_A}^{*,u}$ has free variables in the list $x_{1,m_1}^*, \dots, x_{n,m_n}^*$ while if $t \equiv 0$, its free variables are in list $x_{1,m_1}^*, \dots, x_{n,m_n}^*, u$.

- (ii) If $t \equiv 1$, then x_i occurs free in A if and only if x_{i,m_i}^* occurs free in $A_{\Pi_A}^{*,u}$.
- (iii) If $t \equiv 0$, then x_i occurs free in A if and only if either $s_i \equiv 0$ and $x_{i,m_i}^*[u]$ occurs free in $A_{\Pi_A}^{*,u}$, or $s_i \equiv 1$ and x_{i,m_i}^* occurs free in $A_{\Pi_A}^{*,u}$.
- (iv) The following conditions hold:

- (1) If $\Pi_c \equiv c : \ell \hookrightarrow t$, then

$$c_{\Pi_c}^{*,u} \equiv \begin{cases} c_\ell^*[u], & \text{if } t \equiv 0 \\ c_\ell^*, & \text{if } t \equiv 1 \end{cases}$$

- (2) If $\Pi_x \equiv x : \ell \hookrightarrow t$, then

$$x_{\Pi_x}^{*,u} \equiv \begin{cases} x_\ell^*[u], & \text{if } t \equiv 0 \\ x_\ell^*, & \text{if } t \equiv 1 \end{cases}$$

- (3) If

$$\Pi_{B(C)} \equiv \frac{\begin{array}{cc} \Pi_B & \Pi_C \\ \vdots & \vdots \end{array} \quad \frac{B : \langle \ell_1 \hookrightarrow t_1, \ell_2 \rangle \hookrightarrow t_0 \quad C : \ell_1 \hookrightarrow t_1}{B(C) : \ell_2 \hookrightarrow t}}{B(C) : \ell_2 \hookrightarrow t}$$

then

$$(B(C))_{\Pi_{B(C)}}^{*,u} \equiv \begin{cases} B_{\Pi_B}^{*,u}(C_{\Pi_C}^{*,u}), & \text{if } t \equiv 0, t_0 \equiv 0 \\ B_{\Pi_B}^{*,u}(u, C_{\Pi_C}^{*,u}), & \text{if } t \equiv 0, t_0 \equiv 1 \\ \lambda(u)(B_{\Pi_B}^{*,u}(C_{\Pi_C}^{*,u})), & \text{if } t \equiv 1, t_0 \equiv 0 \\ \lambda(u)(B_{\Pi_B}^{*,u}(u, C_{\Pi_C}^{*,u})), & \text{if } t \equiv 1, t_0 \equiv 1 \end{cases}$$

- (4) If

$$\Pi_{\lambda(v)(B)} \equiv \frac{\begin{array}{c} \Pi_B \\ \vdots \end{array} \quad \frac{B : \ell \hookrightarrow t'}{\lambda(v)(B) : \langle \ell_0 \hookrightarrow t_0, \ell \rangle \hookrightarrow t}}{\lambda(v)(B) : \langle \ell_0 \hookrightarrow t_0, \ell \rangle \hookrightarrow t}$$

then

$$(\lambda(v)(B))_{\Pi_{\lambda(v)B}}^{*,u} \equiv \begin{cases} \lambda(v') \left(B_{\Pi_B}^{*,u} \{v_{\ell_0 \mapsto t_0}^{*,u} := v'\} \right), & \text{if } t \equiv 0, t' \equiv 0 \\ \lambda(v') \left(B_{\Pi_B}^{*,u}(u) \{v_{\ell_0 \mapsto t_0}^{*,u} := v'\} \right), & \text{if } t \equiv 0, t' \equiv 1 \\ \lambda(u) \lambda(v') \left(B_{\Pi_B}^{*,u} \{v_{\ell_0 \mapsto t_0}^{*,u} := v'\} \right), & \text{if } t \equiv 1, t' \equiv 0 \\ \lambda(u) \lambda(v') \left(B_{\Pi_B}^{*,u}(u) \{v_{\ell_0 \mapsto t_0}^{*,u} := v'\} \right), & \text{if } t \equiv 1, t' \equiv 1 \end{cases}$$

where v' is a fresh pure variable of the type of $v_{\ell_0 \mapsto t_0}^{*,u}$.

(5) If

$$\Pi_A \equiv \frac{\begin{array}{ccc} \Pi_{A_0} & \Pi_{A_1} & \Pi_{A_k} \\ \vdots & \vdots & \vdots \\ A_0 : \ell_0 \mapsto t_0 & A_1 : \ell_1 \mapsto t_1 \dots & A_n : \ell_k \mapsto t_k \end{array}}{A_0 \text{ where } \{p_1 := A_1, \dots, p_k := A_k\} : \ell_0 \mapsto t_0}$$

then

$$\begin{aligned} & \left(A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_k\} \right)_{\Pi_A}^{*,u} \\ & \equiv (A_0)_{\Pi_{A_0}}^{*,u} \{ (p_i)_{\ell_i \mapsto t_i}^{*,u} := q_i \mid 1 \leq i \leq k \}, \\ & \text{where} \\ & \left\{ q_1 := (A_1)_{\Pi_{A_1}}^{*,u} \{ (p_i)_{\ell_i \mapsto t_i}^{*,u} := q_i \mid 1 \leq i \leq k \}, \right. \\ & \quad \vdots \\ & \left. q_k := (A_k)_{\Pi_{A_k}}^{*,u} \{ (p_i)_{\ell_i \mapsto t_i}^{*,u} := q_i \mid 1 \leq i \leq k \} \right\}. \end{aligned}$$

where for $i = 1, \dots, n$, q_i is a fresh recursive variable of the type of $(p_i)_{\ell_i \mapsto t_i}^{*,u}$.

Proof. We prove (i)-(iv) simultaneously, by induction on A . Notice that all that needs to be checked are that the transformations defined by (1)-(5) in part (iv) produce terms. If $A \equiv c$ or $A \equiv x$, it is straightforward.

(3) $A \equiv B(C)$. If $t \equiv 1$, then by Proposition 3.2.7, for $i = 1, \dots, n$, x_i is indexed by $m_i \mapsto 1$, and thus, $(x_i)_{m_i \mapsto 1}^{*,u} \equiv x_{i,m_i}^*$.

If $t \equiv 0$ and if, for example, x_i occurs free in B , then by induction hypothesis, if $t_0 \equiv 1$, then $s_i \equiv 1$, and x_{i,m_i}^* occurs free if and only if x_i occurs free and if, on the other hand, $s_0 \equiv 0$, by induction hypothesis,

$x_{i,m_i}^*[u]$ occurs free in $B_{\Pi_B}^{*,u}$ if and only if x_i occurs free in B . Analogous claims can be made if x_i occurs free in C .

(4) $A \equiv \lambda(v)(B)$. Suppose $x_i \neq v$. If $t \equiv 1$, then trivially for $i = 1, \dots, n$, $x_i : m_i \hookrightarrow 1$ and thus, x_{i,m_i}^* occurs free if and only if x_i occurs free. If $t \equiv 0$ and $t' \equiv 1$, then by induction hypothesis for B , x_{i,m_i}^* occurs free. If $t \equiv 0$ and $t' \equiv 0$, then by induction hypothesis, $x_{i,m_i}^*[u]$ occurs free.

(5) $A \equiv A_0$ where $\{p_1 := A_1, \dots, p_k := A_k\}$. The claims are analogous and we do not repeat them here. In general, if for some $i = 1, \dots, k$, $t_i \equiv 1$, then $(p_i)_{\ell_i \hookrightarrow 1}^{*,u} \equiv p_{i,\ell_i}^*$ while, if, $t_i \equiv 0$, then, if p_i occurs free in some A_j ($j = 0, \dots, k$), then by Proposition 3.2.7, it is also the case that $t_j \equiv 0$. Thus, by induction hypothesis, both the variables of $(p_i)_{\ell_i}^{*,u} \equiv p_{i,\ell_i}^*[u]$ occur free if and only if p_i occurs free. It follows that for each i , $\text{rank}(q_i) = \text{rank}(p_i)$ and thus, the recursive term $A_{\Pi_A}^{*,u}$ is well formed. \dashv

If Π_A is a closed locality proof of A with root label $A : \ell \hookrightarrow t$, then we alternatively note its associate term with respect to it and a state variable $u : s$ that does not occur free in it as

$$A_{\Pi_A}^{*,u} \equiv A_{\ell \hookrightarrow t}^{*,u}.$$

This notation will be used when the closed locality proof is explicit and we want to draw attention to the index of the root label instead.

Theorem 4.3.2. *Suppose that the free variables of $A : \tilde{\sigma}$ are in the list $x_1 : \tilde{\sigma}_1, \dots, x_n : \tilde{\sigma}_n$ and the function $f_A : \tilde{\sigma}_1 \times \dots \times \tilde{\sigma}_n \rightarrow \tilde{\sigma}$ is defined by*

$$f_A(f_1, \dots, f_n) = \text{den}(A)(g\{x_1 := f_1, \dots, x_n := f_n\}).$$

(i) *If Π_A is a closed locality proof of A with root label $A : \ell \hookrightarrow 0$ where for $i = 1, \dots, n$, x_i is indexed $m_i \hookrightarrow s_i$ and the function*

$$f_{A_{\ell \hookrightarrow 0}^{*,u}} : s \times \left\{ \begin{array}{c} (s \rightarrow (\tilde{\sigma}_1)_*^{m_1}) \\ (\tilde{\sigma}_1)_*^{m_1} \end{array} \right\} \times \dots \times \left\{ \begin{array}{c} (s \rightarrow (\tilde{\sigma}_n)_*^{m_n}) \\ (\tilde{\sigma}_n)_*^{m_n} \end{array} \right\} \rightarrow (\tilde{\sigma})_*^\ell$$

is defined for $a : s$ by

$$\begin{aligned} f_{A_{\ell \hookrightarrow 0}^{*,u}}(a, h_1, \dots, h_n) \\ = \text{den}(A_{\ell \hookrightarrow 0}^{*,u} \{ (x_1)_{m_1 \hookrightarrow s_1}^{*,u} := x'_1, \dots, (x_n)_{m_n \hookrightarrow s_n}^{*,u} := x'_n \}) \\ (g\{x'_1 := h_1, \dots, x'_n := h_n, u := a\}), \end{aligned}$$

then $f_{A_{\ell \hookrightarrow 0}^{,u}}$ is an associate of f_A with respect to $\langle m_1 \hookrightarrow s_1, \dots, m_n \hookrightarrow s_n, \ell \rangle$.*

(ii) If Π_A is a closed locality proof of A with root label $A : \ell \hookrightarrow 1$ where for $i = 1, \dots, n$, x_i is indexed by $m_i \hookrightarrow s_i$ (with $s_i \equiv 1$) and the function $f_{A_{\ell \hookrightarrow 1}^{*,u}} : \mathbf{s} \times (\mathbf{s} \rightarrow (\tilde{\sigma}_1)_*^{m_1}) \times \dots \times (\mathbf{s} \rightarrow (\tilde{\sigma}_n)_*^{m_n}) \rightarrow (\tilde{\sigma})_*^\ell$ is defined for $a : \mathbf{s}$ by

$$\begin{aligned} & f_{A_{\ell \hookrightarrow 1}^{*,u}}(a, h_1, \dots, h_n) \\ &= \text{den}(A_{\ell \hookrightarrow 1}^{*,u} \{x_{1,m_1}^* := x'_1, \dots, x_{n,m_n}^* := x'_n\}) (g\{x'_1 := h_1, \dots, x'_n := h_n\})(a), \end{aligned}$$

then $f_{A_{\ell \hookrightarrow 1}^{*,u}}$ is an associate of f_A with respect to $\langle m_1 \hookrightarrow 1, \dots, m_n \hookrightarrow 1, \ell \rangle$.

In particular, if A is a closed term and Π_A is a closed locality proof of A with root label $A : \ell \hookrightarrow t$, then the function

$$\begin{aligned} & (a \mapsto \text{den}(A_{\ell \hookrightarrow 0}^{*,u})(g\{u := a\})), \text{ if } t \equiv 0, \text{ or} \\ & \text{den}(A_{\ell \hookrightarrow 1}^{*,u}), \text{ if } t \equiv 1 \end{aligned}$$

is an associate of $\text{den}(A)$ with respect to ℓ .

Proof. We prove (i) and (ii) together by induction on the term A and by appealing to the basic properties of Theorem 2.4.8 just as we appealed to the Theorem 2.2.9 in the proof of the corresponding Theorem 4.2.2. In fact, the proof is basically a direct generalization of that of Theorem 4.2.2, and we include the details only for the sake of completeness.

For each term A , the associate of A in this proof, unless a locality proof is explicitly given, is considered with respect to the locality proofs used in Proposition 4.3.1. It will be also useful to use the following abbreviations:

$$\begin{aligned} g\{x_i := f_i\} &\equiv g\{x_1 := f_1, \dots, x_n := f_n\} \\ g\{x'_i := h_i\} &\equiv g\{x'_1 := h_1, \dots, x'_n := h_n\} \end{aligned}$$

(1) $A \equiv c$. Then, $f_c(f_1, \dots, f_n) = c$. If $t \equiv 1$,

$$f_{c_{\ell \hookrightarrow t}^{*,u}}(a, h_1, \dots, h_n) = \text{den}(c_\ell^*)(g\{x'_i := h_i\})(a) = c_*^\ell(a),$$

and, if $t \equiv 0$,

$$f_{c_{\ell \hookrightarrow t}^{*,u}}(a, h_1, \dots, h_n) = \text{den}(c_\ell^*[u])(g\{x'_i := h_i, u := a\}) = c_*^\ell(a).$$

Thus, by Corollary 2.4.9, in both cases $f_{c_{\ell \hookrightarrow t}^{*,u}}$ is an associate of f_c with respect to $\langle m_1 \hookrightarrow s_1, \dots, m_n \hookrightarrow s_n, \ell \rangle$.

(2) $A \equiv x$. Suppose that for some j , $x_j \equiv x$ and $\Pi_x \equiv x : m_j \hookrightarrow s_j$.

$$f_x(f_1, \dots, f_n) = \text{den}(x)(g\{x_i := f_i\}) = f_j.$$

If $s_j \equiv 1$ and $h_j : s \rightarrow (\tilde{\sigma}_j)_*^{m_j}$,

$$f_{(x)_{m_j \hookrightarrow 1}}^{*,u}(a, h_1, \dots, h_n) = \text{den}(x_{m_j}^* \{x_{m_j}^* := x'_j\})(g\{x'_i := h_i\})(a) = h_j(a)$$

and, if $s_j \equiv 0$ and $h_j : (\tilde{\sigma}_j)_*^{m_j}$,

$$\begin{aligned} f_{(x)_{m_j \hookrightarrow 1}}^{*,u}(a, h_1, \dots, h_n) \\ = \text{den}(x_{m_j}^*[u] \{x_{m_j}^*[u] := x'_j\})(g\{x'_i := h_i, u := a\}) = h_j. \end{aligned}$$

By case (i) of Theorem 2.4.8, in both cases $f_{(x)_{m_j \hookrightarrow s_j}}^{*,u}$ is an associate of f_x with respect to $\langle m_1 \hookrightarrow s_1, \dots, m_j \hookrightarrow s_j, \dots, m_n \hookrightarrow s_n, m_j \rangle$.

(3) $A \equiv B(C)$. Suppose that the free variables of B and C are in the list $x_1 : \tilde{\sigma}_1, \dots, x_n : \tilde{\sigma}_n$.

$$f_{B(C)}(f_1, \dots, f_n) = f_B(f_1, \dots, f_n, f_C(f_1, \dots, f_n)).$$

If $t \equiv 0$ and $t_0 \equiv 0$,

$$\begin{aligned} f_{(B(C))_{\Pi_B(C)}}^{*,u}(a, h_1, \dots, h_n) \\ = \text{den}(B_{\Pi_B}^{*,u} \{(x_1)_{m_1 \hookrightarrow s_1}^{*,u} := x'_1, \dots, (x_n)_{m_n \hookrightarrow s_1}^{*,u} := x'_n\})(g\{x'_i := h_i, u := a\}) \\ \quad \left(\text{den}(C_{\Pi_C}^{*,u} \{(x_1)_{m_1 \hookrightarrow s_1}^{*,u} := x'_1, \dots, (x_n)_{m_n \hookrightarrow s_1}^{*,u} := x'_n\})(g\{x'_i := h_i, u := a\}) \right) \\ = \begin{cases} f_{B_{\Pi_B}}^{*,u} \left(a, h_1, \dots, h_n, f_{C_{\Pi_C}}^{*,u}(a, h_1, \dots, h_n) \right), & \text{if } t_1 \equiv 0, \\ f_{B_{\Pi_B}}^{*,u} \left(a, h_1, \dots, h_n, (a \mapsto f_{C_{\Pi_C}}^{*,u}(a, h_1, \dots, h_n)) \right), & \text{if } t_1 \equiv 1, \end{cases} \end{aligned}$$

if $t \equiv 0$ and $t_0 \equiv 1$,

$$\begin{aligned} f_{(B(C))_{\Pi_B(C)}}^{*,u}(a, h_1, \dots, h_n) \\ = \begin{cases} \left(a \mapsto f_{B_{\Pi_B}}^{*,u}(a, h_1, \dots, h_n) \right) \left(a, f_{C_{\Pi_C}}^{*,u}(a, h_1, \dots, h_n) \right), & \text{if } t_1 \equiv 0, \\ \left(a \mapsto f_{B_{\Pi_B}}^{*,u}(a, h_1, \dots, h_n) \right) \left(a, (a \mapsto f_{C_{\Pi_C}}^{*,u}(a, h_1, \dots, h_n)) \right), & \text{if } t_1 \equiv 1, \end{cases} \end{aligned}$$

if $t \equiv 1$ and $t_0 \equiv 0$,

$$\begin{aligned} f_{(B(C))_{\Pi_B(C)}}^{*,u}(a, h_1, \dots, h_n) \\ = \begin{cases} f_{B_{\Pi_B}}^{*,u}(a, h_1, \dots, h_n, f_{C_{\Pi_C}}^{*,u}(a, h_1, \dots, h_n)), & \text{if } t_1 \equiv 0, \\ f_{B_{\Pi_B}}^{*,u}(a, h_1, \dots, h_n, (a \mapsto f_{C_{\Pi_C}}^{*,u}(a, h_1, \dots, h_n))), & \text{if } t_1 \equiv 1, \end{cases} \end{aligned}$$

and finally, if $t \equiv 1$ and $t_0 \equiv 1$,

$$\begin{aligned} & f_{(B(C))_{\Pi_{B(C)}}^{*,u}}(a, h_1, \dots, h_n) \\ &= \begin{cases} \left(a \mapsto f_{\Pi_B}^{*,u}(a, h_1, \dots, h_n) \right) \left(a, f_{\Pi_C}^{*,u}(a, h_1, \dots, h_n) \right), & \text{if } t_1 \equiv 0, \\ \left(a \mapsto f_{\Pi_B}^{*,u}(a, h_1, \dots, h_n) \right) \left(a, (a \mapsto f_{\Pi_C}^{*,u}(a, h_1, \dots, h_n)) \right), & \text{if } t_1 \equiv 1. \end{cases} \end{aligned}$$

By induction hypothesis, $f_{\Pi_B}^{*,u}$ is an associate of f_B with respect to $\langle m_1 \mapsto s_1, \dots, m_n \mapsto s_n, \ell_1 \mapsto t_1, \ell_2 \rangle$ and $f_{\Pi_C}^{*,u}$ is an associate of f_C with respect to $\langle m_1 \mapsto s_1, \dots, m_n \mapsto s_n, \ell_1 \rangle$. Thus, by the composition case in Theorem 2.4.8, in all four cases the function $f_{(B(C))_{\Pi_{B(C)}}^{*,u}}$ is an associate of $f_{B(C)}$ with respect to $\langle m_1 \mapsto s_1, \dots, m_n \mapsto s_n, \ell_2 \rangle$.

Notice again that the hypotheses of the composition case in Theorem 2.4.8 are met due to Proposition 3.2.7.

(4) $A \equiv \lambda(v)(B)$. Suppose that the free variables of B are in the list $x_1 : \tilde{\sigma}_1, \dots, x_n : \tilde{\sigma}_n, v : \tilde{\sigma}$.

$$f_{\lambda(v)(B)}(f_1, \dots, f_n) = (f \mapsto f_B(f_1, \dots, f_n, f)).$$

If $t \equiv 0$ and $t' \equiv 1$,

$$\begin{aligned} & f_{(\lambda(v)(B))_{\Pi_{\lambda(v)B}}^{*,u}}(a, h_1, \dots, h_n) \\ &= \text{den}(\lambda(v')B_{\Pi_B}^{*,u}(u) \{v_{\ell_0 \mapsto t_0}^{*,u} := v', (x_1)_{m_1 \mapsto s_1}^{*,u} := x'_1, \dots, (x_n)_{m_n \mapsto s_1}^{*,u} := x'_n\}) \\ & \quad (g\{x'_i := h_i, u := a\}) \\ &= \left(h \mapsto \text{den}(B_{\Pi_B}^{*,u}(u) \{v_{\ell_0 \mapsto t_0}^{*,u} := v', (x_1)_{m_1 \mapsto s_1}^{*,u} := x'_1, \dots, (x_n)_{m_n \mapsto s_1}^{*,u} := x'_n\}) \right. \\ & \quad \left. (g\{x'_i := h_i, u := a, v' := h\}) \right), \\ &= \left(h \mapsto (b \mapsto f_{\Pi_B}^{*,u}(b, h_1, \dots, h_n, h))(a) \right). \end{aligned}$$

If $t \equiv 0$ and $t' \equiv 0$,

$$\begin{aligned}
& f_{(\lambda(v)(B))_{\Pi_{\lambda(v)B}}^{*,u}}(a, h_1, \dots, h_n) \\
&= \text{den}(\lambda(v')B_{\Pi_B}^{*,u}\{v_{\ell_0 \mapsto t_0}^{*,u} := v', (x_1)_{m_1 \mapsto s_1}^{*,u} := x'_1, \dots, (x_n)_{m_n \mapsto s_n}^{*,u} := x'_n\}) \\
&\quad (g\{x'_i := h_i, u := a\}) \\
&= \left(h \mapsto \text{den}(B_{\Pi_B}^{*,u}\{v_{\ell_0 \mapsto t_0}^{*,u} := v', (x_1)_{m_1 \mapsto s_1}^{*,u} := x'_1, \dots, (x_n)_{m_n \mapsto s_n}^{*,u} := x'_n\}) \right. \\
&\quad \left. (g\{x'_i := h_i, u := a, v' := h\}) \right) \\
&= \left(h \mapsto f_{B_{\Pi_B}^{*,u}}(a, h_1, \dots, h_n, h) \right).
\end{aligned}$$

and if $t \equiv 1$, similarly,

$$\begin{aligned}
& f_{(\lambda(v)(B))_{\Pi_{\lambda(v)B}}^{*,u}}(a, h_1, \dots, h_n) \\
&= \begin{cases} \left(b \mapsto \left(h \mapsto \left(b \mapsto f_{B_{\Pi_B}^{*,u}}(b, h_1, \dots, h_n, h) \right) (b) \right) \right) (a), & \text{if } t' \equiv 1, \\ \left(b \mapsto \left(h \mapsto f_{B_{\Pi_B}^{*,u}}(b, h_1, \dots, h_n, h) \right) \right) (a) & \text{if } t' \equiv 0. \end{cases}
\end{aligned}$$

By induction hypothesis, the function $f_{B_{\Pi_B}^{*,u}}$ is an associate of f_B with respect to $\langle m_1 \mapsto s_1, \dots, m_n \mapsto s_n, \ell \rangle$ and, thus, by the λ -abstraction case in Theorem 2.4.8, the function $f_{(\lambda(v)(B))_{\Pi_{\lambda(v)B}}^{*,u}}$ is an associate of $f_{\lambda(v)(B)}$ with respect to $\langle m_1 \mapsto s_1, \dots, m_n \mapsto s_n, \ell_0 \mapsto t_0, \ell \rangle$.

(5) $A \equiv A_0$ where $\{p_1 := A_1, \dots, p_k := A_k\}$. Suppose that the free variables of A are in the list x_1, \dots, x_n and thus, the free variables of each $A_i (i = 1, \dots, k)$ are in the list $x_1, \dots, x_n, p_1, \dots, p_k$.

$$f_A(f_1, \dots, f_n) = f_{A_0}(f_1, \dots, f_n, P_1, \dots, P_k),$$

where for $i = 1, \dots, k$, each

$$P_i = f_{P_i}(f_1, \dots, f_n)$$

is defined by recursion on the $\text{rank}(p_i)$:

$$P_i = f_{A_i}(f_1, \dots, f_n, P_{r_1}, \dots, P_{r_m})$$

where p_{r_1}, \dots, p_{r_m} are the variables with $\text{rank} < \text{rank}(p_i)$ (see also Section 1.5.1).

On the other hand, since A and A_0 have the same output index, whether $t_0 \equiv 0$ or $t_0 \equiv 1$,

$$f_{A_{\Pi_A}^{*,u}}(a, h_1, \dots, h_n) = f_{(A_0)_{\Pi_{A_0}}^{*,u}}(a, h_1, \dots, h_n, Q_1, \dots, Q_m)$$

where, in each case, h_1, \dots, h_n may be of different types and Q_i is defined recursively on the $\text{rank}(q_i)$ by

$$Q_i = f_{Q_i}(a, h_1, \dots, h_n) = f_{(A_i)_{\ell_i \mapsto t_i}^{*,u}}(a, h_1, \dots, h_n, Q_{r_1}, \dots, Q_{r_m}).$$

It is important to notice again here that by Proposition 4.3.1, for $i = 1, \dots, k$, $\text{rank}(p_i) = \text{rank}(q_i)$.

The proof is concluded by an easy induction on the $\text{rank}(p_i)$, using the composition case of Theorem 2.4.8, according to which for $i = 1, \dots, k$, f_{Q_i} is an associate of f_{P_i} with respect to ℓ_i . Notice that the restriction of the Theorem is guaranteed by Proposition 3.2.7. \dashv

Finally, the following lemma generalizes Lemma 4.2.3 of the previous section.

Lemma 4.3.3. *If $A : \tilde{\sigma}$ is an irreducible term and Π_A is a closed locality proof of A with root label $A : \ell \mapsto t$, then the term $A_{\Pi_A}^{*,u}$ is an irreducible term.*

Proof. The proof is by a straightforward, but rather tedious, induction on A . \dashv

Chapter 5

Algorithmic Factual Content

The aim of the work presented in this thesis is to define in the Theory of Referential Intentions a notion of structural meaning which represents what a sentence communicates about the world at a particular context of reference. A key idea of the approach adopted here was to study the *locality behavior* of objects which is coded by *closed locality input indices*. In Chapter 2, we showed that if an object $f : \tilde{\sigma}$ has an *associate* with respect to a closed locality input index ℓ of its type, then its behavior is characterized completely by it. In Chapter 3, we described the locality behavior of state-dependent terms of $\mathbf{L}_{\text{ar}}^\lambda$ with the use of the *closed locality proofs* and showed that for any term A , its locality behavior is determined by the locality behavior of the constants that occur in it. The main technical result of the thesis was also presented in this chapter: for every term A , there is a most local closed locality proof. To be more precise, for every term A such that the locality behavior of the constants that occur in it is “described in a canonical way” (by a *generic locality proof schema*), there is a behavior of the denotation of A based on which its computation at a state uses the values of the denotations of the subterms at that particular state *as much as possible*. Then, in Chapter 4, for any term A and a closed locality proof of A , we defined formally with the use of the *associate transformation* in an extension of $\mathbf{L}_{\text{ar}}^\lambda$ the associate of A with respect to the given locality proof.

In this last chapter of the thesis, we use the formal associate of a term A with respect to its most local locality proof at a state a to define the **factual content of A at a** (Section 5.1). We also define **factual synonymy** and we use examples to shed some light on the role of the newly introduced notion of meaning in natural language semantics. In Section 5.2, we compare the proposed notions with the corresponding ideas in the Logic of Demonstra-

tives and finally, in Section 5.3, we present some ideas about future work in this area.

5.1 Factual Canonical Form and Factual Synonymy

In $\mathbb{L}_{\text{ar}}^\lambda$, local meaning and local synonymy are defined only for terms of type $\tilde{\mathfrak{t}}$, that is, terms that render natural language sentences (Section 1.5.2). We define here analogously factual content for all terms of type $A : \tilde{\sigma}_0$ and we use it to investigate the synonymy of utterances.

Definition 5.1.1. (Factual Content of a Term $A : \tilde{\sigma}_0$ at a State a) Let $A : \tilde{\sigma}_0$ and a state $a : \mathfrak{s}$ in $\mathbb{L}_{\text{ar}}^\lambda$ be such that

$$A \Rightarrow_{\text{cf}} A_0 \text{ where } \{p_1 := A_1, \dots, p_n := A_n\}.$$

If Π_A is the most local locality proof of $\text{cf}(A)$ with root label $\text{cf}(A) : l \Downarrow 0$, then the *factual canonical form of A at state a* is

$$\text{fcf}(A, \bar{a}) := (\text{cf}(A))_{\Pi_A}^{*, \bar{a}}$$

and the *factual content of A at state a* is the referential intension of the factual canonical form of A at a , that is,

$$\text{Factual Content of } A \text{ at state } a \equiv \text{int}(\text{fcf}(A, \bar{a})).$$

As usual, we will use the factual canonical form of A at a state a to talk about its factual content syntactically.

Naturally enough, the factual canonical form of a local term $A : \tilde{\sigma}_0$ at state a is simply

$$\text{fcf}(A, \bar{a}) := (\text{cf}(A))^{*, \bar{a}}.$$

Also, by Lemma 4.3.3, the term $(\text{cf}(A))_{\Pi_A}^{*, \bar{a}}$ is irreducible and thus, its canonical form is congruent to it.

Second, notice that for any term $A : \tilde{\sigma}$, the output index of the root label of its most local locality proof is 0. (It follows easily since, by induction on A , it can be showed that if A has a closed locality proof with root label $\ell \Downarrow 1$, then there is also a closed locality proof of A with root label $\ell \Downarrow 0$.)

The notion of factual synonymy is defined naturally as referential synonymy between factual canonical forms.

Definition 5.1.2. (Factual Synonymy) For any two terms $A : \tilde{\sigma}_0$ and $B : \tilde{\sigma}_0$, A at state a is *factually synonymous* with B at state b if and only if

$$\text{fcf}(A, \bar{a}) \approx \text{fcf}(B, \bar{b}).$$

Example 5.1.3. Consider the famous puzzle, discussed in [22], about the two utterances ‘He is Scott’ and ‘Scott is Scott’ at a state where indeed ‘he’ and ‘Scott’ refer to the same person. We assume a local constant $\text{id} : \tilde{\mathbf{e}} \times \tilde{\mathbf{e}} \rightarrow \tilde{\mathbf{t}}$ that stands for equality between terms of type $\tilde{\mathbf{e}}$, and thus, the two sentences are rendered in $\mathbf{L}_{\text{ar}}^\lambda$ by the local terms

$$\text{id}(\text{he}, \text{Scott}) \quad \text{and} \quad \text{id}(\text{Scott}, \text{Scott}).$$

Their respective factual contents at state a where $he(a) = \text{Scott}(a)$, are defined by the corresponding canonical forms and their factual synonymy follows easily.

$$\begin{aligned} \text{id}^*[\bar{a}](p, q) \text{ where } \{p := \text{he}^*[\bar{a}], q := \text{Scott}^*[\bar{a}]\} \\ \approx \text{id}^*[\bar{a}](p, q) \text{ where } \{p := \text{Scott}^*[\bar{a}], q := \text{Scott}^*[\bar{a}]\}. \end{aligned}$$

Notice that in determining factual synonymy only the values of the denotations of he and $Scott$ at that particular state are used exactly because the corresponding constants are local. The two utterances communicate the same fact about the world which explains why we expect someone to treat them as “synonymous”.

On the other hand, their factual synonymy rests on an equivalence between two denotations ($\text{he}^*[\bar{a}] = \text{Scott}^*[\bar{a}]$) and goes beyond the knowledge of the language. It requires knowledge of the external (disguised) world which explains the possibility of having different attitudes towards them — for example, believing one and disbelieving the other. In [22], it is also explained that these two sentences which, of course, are not referentially synonymous, are also not locally synonymous at a

$$\begin{aligned} \text{id}(p', q')(\bar{a}) \text{ where } \{p' := \text{he}, q' := \text{Scott}\} \\ \not\approx \text{id}(p', q')(\bar{a}) \text{ where } \{p' := \text{Scott}, q' := \text{Scott}\}. \end{aligned}$$

This simple example shows that the factual content of an utterance cannot be the object of belief although as a notion of situated meaning it is an obvious candidate. In [22] and [13], it is suggested that maybe local meanings are more suitable for that. It is more important though to point out the fact that in $\mathbf{L}_{\text{ar}}^\lambda$ there are now two notions of situated meaning that express a different algorithm of computation of the denotation of a term at a state and thus, can possibly contribute to our understanding of propositional attitudes in general.

$$\begin{array}{c}
\text{former} : \langle \langle l \mapsto 0 \rangle \mapsto 1, l \mapsto 0 \rangle \mapsto 0 \quad p_1 : \langle l \mapsto 0 \rangle \mapsto 1 \\
\hline
\text{former}(p_1) : \langle l \mapsto 0 \rangle \mapsto 0 \quad p_2 : l \mapsto 0 \\
\hline
\text{former}(p_1, p_2) : l \mapsto 0 \quad \text{minister} : \langle l \mapsto 0 \rangle \mapsto 1 \quad \text{John} : l \mapsto 0 \\
\hline
\text{former}(p_1, p_2) \text{ where } \{p_1 := \text{minister}, p_2 := \text{John}\} : l \mapsto 0
\end{array}$$

Figure 5.1: The most local locality proof of $\text{cf}(\text{former}(\text{minister}, \text{John}))$.

Example 5.1.4. Consider another example with the indexical ‘he’ and the adjective ‘former’, whose locality behavior we explained many times (Examples 2.3.1 and 3.2.10), in the two utterances ‘John is a former minister’ and ‘He is a former minister’ at a state a where $\text{John}(a) = \text{he}(a)$. The most local locality proof of the canonical form of $\text{former}(\text{minister}, \text{John})$ is shown in Figure 5.1. The most local proof of the canonical form of $\text{former}(\text{minister}, \text{he})$ is completely analogous and thus, it follows that

$$\begin{aligned}
& \text{former}^*_{\langle \langle l \mapsto 0 \rangle \mapsto 1, l \mapsto 0 \rangle}[\bar{a}](p, q) \text{ where } \{p := \text{minister}^*_{\langle l \mapsto 0 \rangle}, q := \text{John}^*_l[\bar{a}]\} \\
& \approx \text{former}^*_{\langle \langle l \mapsto 0 \rangle \mapsto 1, l \mapsto 0 \rangle}[\bar{a}](p, q) \text{ where } \{p := \text{minister}^*_{\langle l \mapsto 0 \rangle}, q := \text{he}^*_l[\bar{a}]\}.
\end{aligned}$$

These two utterances are not locally synonymous but the factual synonymy result agrees again with our intuition that these two utterances convey the same information about the world. Examples involving indexical expressions are very useful in motivating the need of factual synonymy and in what follows we will investigate another one.

Example 5.1.5. We formulate in $\mathbb{L}_{\text{ar}}^\lambda$ the simple Fregean example that we have already mentioned in Section 1.1 which involves the indexical expressions ‘Yesterday’ and ‘Today’. An utterance of ‘It is raining today’ at a particular day, at state a , will be reported as ‘It rained yesterday’ the next day, at state b . The corresponding terms in $\mathbb{L}_{\text{ar}}^\lambda$ and their canonical forms are shown below¹.

$$\begin{aligned}
& \text{Yesterday}(\text{it_rains}) \Rightarrow_{\text{cf}} \text{Yesterday}(p) \text{ where } \{p := \text{it_rains}\} \\
& \text{Today}(\text{it_rains}) \Rightarrow_{\text{cf}} \text{Today}(p) \text{ where } \{p := \text{it_rains}\}
\end{aligned}$$

The most local locality proof of ‘It rained yesterday’ is shown in Figure 5.2. Under the assumption that the constant **Today** has also only the

¹The constants $\text{Yesterday} : \tilde{\mathfrak{t}} \rightarrow \tilde{\mathfrak{t}}$ and $\text{Today} : \tilde{\mathfrak{t}} \rightarrow \tilde{\mathfrak{t}}$ are treated in this example as sentential operators (see also the treatment of ‘Yesterday’ in Sections 1.6 and 5.2).

$$\frac{\text{Yesterday} : \langle l \mapsto 1 \rangle \mapsto 0 \quad p : l \mapsto 1}{\frac{\text{Yesterday}(p) : l \mapsto 0}{\text{Yesterday}(p) \text{ where } \{p := \text{it_rains}\} : l \mapsto 0} \quad \text{it_rains} : l \mapsto 1}$$

Figure 5.2: The most local closed locality proof of the canonical form of $\text{Yesterday}(\text{it_rains})$.

closed locality index $\langle l \mapsto 1 \rangle$, then the corresponding proof of the canonical form of ‘It is raining today’ is exactly the same if we simply replace the constant **Yesterday** with **Today**. The two utterances are factually synonymous

$$\begin{aligned} & \text{Yesterday}^*_{\langle l \mapsto 1 \rangle}[\bar{b}](p') \text{ where } \{p' := \text{it_rains}_l^*\} \\ & \approx \text{Today}^*_{\langle l \mapsto 1 \rangle}[\bar{a}](p') \text{ where } \{p' := \text{it_rains}_l^*\} \end{aligned}$$

provided that

$$\models \text{Yesterday}^*_{\langle l \mapsto 1 \rangle}[\bar{b}](p') = \text{Today}^*_{\langle l \mapsto 1 \rangle}[\bar{a}](p')$$

which follows since by the natural interpretation of these constants

$$\models \text{Yesterday}(p')(\bar{b}) = \text{Today}(p')(\bar{a}).$$

On the other hand, if we consider that the constant **Today** is local, then the two utterances are not synonymous. Notice that the two utterances are locally synonymous.

In general, reported speech, as in the example above, is a phenomenon that we are tempted to analyze with the use of factual synonymy. Notice though that, factual synonymy (like referential and local synonymy) is structural which means that a sentence in reported speech will be factually synonymous with the original one only if its constituent parts are reported accordingly.

Example 5.1.6. We end this series of examples with the famous Partee example ([18]) which we try to analyze with the use of locality as explained here. The canonical form of the sentence ‘The temperature is ninety and rising’ is

$$\begin{aligned} & \lambda(x)(\text{and}(\text{rise}(x), \text{ninety}(x)))(\text{the}(\text{temp})) \\ & \Rightarrow_{\text{cf}} \lambda(x)(\text{and}(r_1(x), r_2(x)))(p) \text{ where } \begin{cases} r_1 := \lambda(x)\text{rise}(x), \\ r_2 := \lambda(x)\text{ninety}(x), \\ p := \text{the}(q), q := \text{temp} \end{cases}. \end{aligned}$$

$$\begin{array}{c}
\frac{\frac{r_1 : \langle l \mapsto 1 \rangle \mapsto 0 \quad x : l \mapsto 1}{\text{and} : \langle l \mapsto 0, l \mapsto 0 \rangle \mapsto 0} \quad \frac{r_1(x) : l \mapsto 0}{r_1(x) : l \mapsto 0} \quad \frac{r_2 : \langle l \mapsto 1 \rangle \mapsto 0 \quad x : l \mapsto 1}{r_2(x) : l \mapsto 0}}{\text{and}(r_1(x)) : \langle l \mapsto 0 \rangle \mapsto 0} \\
\\
\Pi_{A_0} : \frac{\frac{\text{and}(r_1(x), r_2(x)) : l \mapsto 0}{\lambda(x)(\text{and}(r_1(x), r_2(x))) : \langle l \mapsto 1 \rangle \mapsto 0} \quad p : l \mapsto 1}{\lambda(x)(\text{and}(r_1(x), r_2(x)))(p) : l \mapsto 0} \\
\\
\Pi_{A_{r_1}} : \frac{\frac{\text{rise} : \langle l \mapsto 1 \rangle \mapsto 0 \quad x : l \mapsto 1}{\text{rise}(x) : l \mapsto 0}}{\lambda(x)\text{rise}(x) : \langle l \mapsto 1 \rangle \mapsto 0} \quad \Pi_{A_{r_2}} : \frac{\frac{\text{ninety} : \langle l \mapsto 1 \rangle \mapsto 0 \quad x : l \mapsto 1}{\text{ninety}(x) : l \mapsto 0}}{\lambda(x)\text{ninety}(x) : \langle l \mapsto 1 \rangle \mapsto 0} \\
\\
\Pi_{A_p} : \frac{\text{the} : \langle \langle l \mapsto 0 \rangle \mapsto 1 \rangle \mapsto 0 \quad q : \langle l \mapsto 0 \rangle \mapsto 1}{\text{the}(q) : l \mapsto 1} \quad \Pi_{A_q} : \text{temp} : \langle l \mapsto 0 \rangle \mapsto 1 \\
\\
\frac{\Pi_{A_0} \quad \Pi_{A_{r_1}} \quad \Pi_{A_{r_2}} \quad \Pi_{A_p} \quad \Pi_{A_q}}{A}
\end{array}$$

Figure 5.3: The most local closed locality proof of the term $A \equiv \text{cf}(\lambda(x)(\text{and}(\text{rise}(x), \text{ninety}(x)))(\text{the}(\text{temp})))$.

Its most local closed locality proof (Figure 5.3) is shown here in order to make clear how the coordination of *rise* and *ninety* forces the use of the index $\langle l \mapsto 1 \rangle$ of the local constant *ninety* (see also Example 3.2.6). Its factual content at a state a is

$$\begin{aligned}
& \lambda(x')(\text{and}_{\langle l \mapsto 0, l \mapsto 0 \rangle}^*[\bar{a}](r'_1(x'), r'_2(x'))(p')) \\
& \text{where } \{r'_1 := \lambda(x')\text{rise}_{\langle l \mapsto 1 \rangle}^*[\bar{a}](x'), \\
& \quad r'_2 := \lambda(x')\text{ninety}_{\langle l \mapsto 1 \rangle}^*[\bar{a}](x'), \\
& \quad p' := \lambda(u)\text{the}_{\langle \langle l \mapsto 0 \rangle \mapsto 1 \rangle}^*[u](q'), \\
& \quad q' := \text{temp}_{\langle l \mapsto 0 \rangle}^*\}.
\end{aligned}$$

Thus, its factual content at any state a depends on the associate of the subterm *the(temp)* and not just its value at a . This is *entailed* here by the analysis of the locality behavior of the particular sentence; it is not imposed by choices made in rendering the sentence formally.

The analysis of the locality behavior of the sentence ‘The temperature is ninety and it is rising’ if we use co-indexing as in [22] will give completely analogous results for the subterm **the(temp)**.

$$\begin{aligned} &\text{The temperature is ninety and it is rising} \\ &\xrightarrow{\text{render}} \text{and}(p, q) \text{ where } \{p := \text{ninety}(p_1), q := \text{rise}(p_1), \\ &\quad p_1 := \text{the}(p_2), p_2 := \text{temp}\}. \end{aligned}$$

On the other hand, the analysis of the locality behavior of the sentence ‘The temperature is ninety and the temperature is rising’ considers the two occurrences of the subterm **the(temp)** independently.

$$\begin{aligned} &\text{and}(\text{ninety}(\text{the}(\text{temp})), \text{rise}(\text{the}(\text{temp}))) \\ &\Rightarrow_{\text{cf}} \text{and}(p, q) \text{ where } \{p := \text{ninety}(p_1), q := \text{rise}(q_1), p_1 := \text{the}(p_2), p_2 := \text{temp}, \\ &\quad q_1 := \text{the}(q_2), q_2 := \text{temp}\}. \end{aligned}$$

In the most local locality proof of this term the subterm **the(temp)** has different locality output indices. In general, a constant that occurs in two places in a term A may have a different locality index in each of its occurrences in the most local locality proof of A .

5.2 Factual Content and the Logic of Demonstratives

In this brief section, we examine some examples that were mentioned already in Section 1.4 where the Logic of Demonstratives was introduced. It is interesting to compare the treatment of these examples in $\mathbf{L}_{\text{ar}}^\lambda$ with that in \mathbf{Ty}_2 which was presented in Section 1.6.

Example 5.2.1. Consider the utterance ‘I was insulted yesterday’ at two states a and b where the agent differs. Kaplan correctly argues that the corresponding utterances don’t have the same Content at these states, although their denotations might be the same.

In $\mathbf{L}_{\text{ar}}^\lambda$, the canonical form of the term that renders the sentence is

$$\text{Yesterday}_1(\text{be_insulted}, l) \Rightarrow_{\text{cf}} \text{Yesterday}_1(p, q) \text{ where } \{p := \text{be_insulted}, q := l\}$$

and its factual canonical form at a state a computed by its most local locality proof shown in Figure 5.4 is

$$(\text{Yesterday}_1)^*_{\langle\langle l \mapsto 0 \rangle \mapsto 1, l \mapsto 0 \rangle}[\bar{a}](p', q') \text{ where } \{p' := \text{be_insulted}^*_{\langle l \mapsto 0 \rangle}, q' := l^*[\bar{a}]\}.$$

$$\begin{array}{c}
\text{Yesterday}_1 : \langle \langle l \mapsto 0 \rangle \mapsto 1, l \mapsto 0 \rangle \mapsto 0 \quad p : \langle l \mapsto 0 \rangle \mapsto 1 \\
\hline
\text{Yesterday}_1(p) : \langle l \mapsto 0 \rangle \mapsto 0 \quad q : l \mapsto 0 \\
\hline
\text{Yesterday}_1(p, q) : l \mapsto 0 \quad \text{be_insulted} : \langle l \mapsto 0 \rangle \mapsto 1 \quad l : l \mapsto 0 \\
\hline
\text{Yesterday}_1(p, q) \text{ where } \{p := \text{be_insulted}, q := l\} : l \mapsto 0
\end{array}$$

Figure 5.4: The most local closed locality proof of the canonical form of $\text{Yesterday}_1(\text{insulted}, l)$.

It is straightforward that if $\not\models l_i^*[\bar{a}] = l_i^*[\bar{b}]$, then

$$\begin{aligned}
& (\text{Yesterday}_1)^*_{\langle \langle l \mapsto 0 \rangle \mapsto 1, l \mapsto 0 \rangle}[\bar{a}](p', q') \text{ where } \{p' := \text{be_insulted}^*_{\langle l \mapsto 0 \rangle}, q' := l_i^*[\bar{a}]\} \\
& \approx (\text{Yesterday}_1)^*_{\langle \langle l \mapsto 0 \rangle \mapsto 1, l \mapsto 0 \rangle}[\bar{b}](p', q') \text{ where } \{p' := \text{be_insulted}^*_{\langle l \mapsto 0 \rangle}, q' := l_i^*[\bar{b}]\}.
\end{aligned}$$

It can be shown that these two utterances are never locally synonymous if a is not equal to b .

Example 5.2.2. Now, consider a state a where the date is the 21st April 1973 and the agent is the individual denoted by the name ‘David Kaplan’. Then the utterance of ‘I was insulted yesterday’ at state a is factually synonymous with the utterance of ‘David Kaplan is insulted on 20 April 1973’ at any state b , provided that the constant DavidKaplan_i^* denotes at b the same individual,

$$\begin{aligned}
& (\text{Yesterday}_1)^*_{\langle \langle l \mapsto 0 \rangle \mapsto 1, l \mapsto 0 \rangle}[\bar{a}](p', q') \text{ where } \{p' := \text{be_insulted}^*_{\langle l \mapsto 0 \rangle}, q' := l_i^*[\bar{a}]\} \\
& \approx (\text{on20April1973})^*_{\langle \langle l \mapsto 0 \rangle \mapsto 1, l \mapsto 0 \rangle}[\bar{b}](p', q') \text{ where} \\
& \quad \{p' := \text{be_insulted}^*_{\langle l \mapsto 0 \rangle}, q' := \text{DavidKaplan}_i^*[\bar{b}]\}.
\end{aligned}$$

Kaplan argues that naturally, these two sentences do not have the same Character. In L_{ar}^λ , these two terms are not referentially synonymous but also the two utterances at state a and b respectively are not locally synonymous. Thus, the remark of Kaplan that ‘I was insulted yesterday’ at a “...will have a content roughly equivalent to...” ([14]) the utterance of ‘David Kaplan is insulted on 20 April 1973’ at any state is made here completely precise.

Finally, we have to point out again the fact that in the Theory of Referential Intentions both factual content and local meaning are structural notions of meaning. Thus, the utterance of ‘I was insulted yesterday’ at state a won’t be factually synonymous with any utterance of ‘The man with the grey hat was insulted yesterday’ at the same state even when the agent happens to be the only man present who wears a grey hat.

5.3 Future Work

The Theory of Referential Intentions provides a mathematical framework of natural language semantics which is now equipped with an additional notion of situated meaning — the factual content. There are various aspects of it that are interesting enough to be investigated further and thus, help us understand better natural language phenomena.

First, the factual content of a term A at a state a is defined with the use of the formal associate of the canonical form of A rather than the formal associate of A itself. If A is a local term, it can be shown that the canonical form of the local associate term of A at a state a is congruent with the local associate term of the canonical form of A at a . In the general case, it is a matter of further investigation to establish the exact relation of the two terms. Part of the complication of this task is the comparison between the most local closed locality proof of A and the most local locality proof of the canonical form of A .

Second, consider two terms A and B that render natural language sentences. If they are referentially synonymous, then they are locally synonymous at any state a (see also [13] for analogous results in $\mathcal{L}_{ar}^{\lambda, G}$). The question whether the two terms are also factually synonymous at any state a is not straightforward to answer. The factual content as it is defined here expresses an algorithm that is based on the most local interpretation of the parts of a term. On the other hand, local meaning expresses a different algorithm that computes the same object but does not make any kind of assumption on the locality behavior of the parts of the term. Thus, the question is really more complicated, and at the same time, more interesting, than it appears at first sight.

Third, one can investigate the role of the meaning that is defined by considering the function over states of the factual content of a term A at a state and how it is related to the global meaning originally defined in $\mathcal{L}_{ar}^{\lambda}$.

Finally, we think that the notion of locality that is introduced here and the analysis with respect to it that is accomplished by the most local locality proof of a sentence can be a useful tool in understanding oblique contexts and related compositionality issues.

Bibliography

- [1] Rudolf Carnap. *Meaning and Necessity*. The University of Chicago Press, second edition, 1956. First Edition in 1947.
- [2] David D. Dowty, Robert E. Wall, and Stanley Peters. *Introduction to Montague semantics*. D. Reidel Publishing Company, 1981.
- [3] Gottlob Frege. Funktion und Begriff. Address given to the *Jenaische Gesellschaft für Medizin und Naturwissenschaft*, Jena, January 9, 1891. English translation by P. T. Geach as ‘Function and Concept’ in [9].
- [4] Gottlob Frege. *Die Grundlangen der Arithmetik: eine logisch-mathematische Untersuchung über den Begriff der Zahl*. Breslau: W. Koebner, 1884.
- [5] Gottlob Frege. Über Begriff und Gegenstand. *Vierteljahrsschrift für wissenschaftliche Philosophie*, 16:192–205, 1892. English translation by P. T. Geach as ‘On Concept and Object’ in [9].
- [6] Gottlob Frege. Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik*, 100:25–50, 1892. English translation by M. Black as ‘On Sense and Meaning’ in [9].
- [7] Gottlob Frege. Der Gedanke - Eine logische Untersuchung. *Beiträge zur Philosophie des deutschen Idealismus I*, pages 58–77, 1918-1919. English translation by P. Geach and R. H. Stoothoff as ‘Thoughts’, Reprinted in [25].
- [8] Daniel Gallin. *Intensional and higher-order modal logic (With applications to Montague semantics)*. Number 19 in North-Holland Mathematical Studies. North-Holland, 1975.
- [9] P. Geach and M. Black, editors. *Translations from the philosophical writings of Gottlob Frege*. Basil Blackwell, third edition, 1980.

- [10] Irene Heim and Angelika Kratzer. *Semantics in Generative Grammar*. Blackwell, 1998.
- [11] Theo M.V. Janssen. Frege, Contextuality and Compositionality. *Journal of Logic, Language, and Information*, 10:115–136, 2001.
- [12] Eleni Kalyvianaki. Factual content in algorithmic natural language semantics. In Ville Nurmi and Dmitry Sustretov, editors, *Proceedings of the Twelfth ESSLLI Student Session*, 2007.
- [13] Eleni Kalyvianaki and Yiannis N. Moschovakis. Two aspects of situated meaning. In Fritz Hamm and Stephan Kepser, editors, *Logics for Linguistic Structures*. DeGruyter, To appear.
- [14] David Kaplan. On the logic of demonstratives. *Journal of Philosophical Logic*, 8:81–98, 1978. Reprinted in [25].
- [15] David Kaplan. Demonstratives An Essay on the Semantics, Logic, Metaphysics, and Epistemology of Demonstratives and Other Indexicals & Afterthoughts. In Joseph Almog, John Perry, and Howard Wettstein, editors, *Themes from Kaplan*, pages 481–614. Oxford University Press, 1989.
- [16] Richard Montague. English as a formal language. In Bruno Visentini et al., editor, *Linguaggi nella Società e nella Tecnica*, pages 189–224. Milan, Edizioni di Comunità, 1970. Reprinted in [19].
- [17] Richard Montague. Universal grammar. *Theoria*, (36):373–398, 1970. Reprinted in [19].
- [18] Richard Montague. The proper treatment of quantification in ordinary english. In J. Hintikka, J. Moravcsik, and P. Suppes, editors, *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*, pages 221–242. Dordrecht, D. Reidel Publishing Company, 1973. Reprinted in [19].
- [19] Richard Montague. *Formal Philosophy: Selected papers by Richard Montague*. Richmond H. Thomason (editor), Yale University Press, 1974.
- [20] Yiannis N. Moschovakis. Sense and denotation as algorithm and value. In J. Väänänen and J. Oikkonen, editors, *Logic Colloquium '90*, volume 2 of *Lecture Notes in Logic*, pages 210–249, 1994.

- [21] Yiannis N. Moschovakis. What is an algorithm? In B. Engquist and W. Schmid, editors, *Mathematics unlimited — 2001 and beyond*, pages 919–936. Springer, 2001.
- [22] Yiannis N. Moschovakis. A logical calculus of meaning and synonymy. *Linguistics and Philosophy*, 29:27–89, 2006.
- [23] Francis Jeffry Pelletier. Did Frege believe Frege’s principle? *Journal of Logic, Language, and Information*, 10:87–114, 2001.
- [24] Paul Portner and Barbara H. Partee, editors. *Formal Semantics: The essential readings*. Blackwell, 2002.
- [25] Nathan Salmon and Scott Soames, editors. *Propositions and Attitudes*. Oxford University Press, 1988.
- [26] Dana Scott. Advice on Modal Logic. In Karel Lambert, editor, *Philosophical problems in logic: some recent developments*, pages 143–173. D. Reidel: Dordrecht, Holland, 1970.

Index

- IL, intensional logic, 10
- λ -abstraction, 9
- λ -calculus, 8
- L_{ar}^λ , 18
- LD, logic of demonstratives, 16, 103
- LIL, language of intensional logic, 12

- acyclic recursion, 19
- application, 9
 - associate, 80
- assignment, 10
 - update, 10
- associate
 - application, 80
 - term, 87
 - transformation, 87
 - transformation, local, 82
 - type, 44
- associate (function), 45
 - formal, 87
 - local, 33
 - formal, 82
 - preferred, 36
 - preferred, 48

- basic type, 8

- canonical form, 21
 - factual, 98
- character, in LD, 17
- circumstance, possible, 16
- closed
 - locality index, 40
 - locality input index, 40
 - locality output index, 40
 - locality proof, 58
 - term, 9

- compositionality principle, 1
- congruence, 19
- constant, 9
- constant, (locality) index, 40
- content
 - factual, 2, 98
 - in LIL, 28
 - in LD, 17
- context
 - of evaluation, 16
 - of utterance, 16
 - principle, 7

- declarative sentence, 6
- demonstrative, 16
 - logic of, LD, 16, 103
- denotation, 6, 10

- equivalent (set of conditions), 71
- evaluation, 42
- explicit term, 23
- extension, 11
 - Montague, 14

- factual
 - canonical form, 98
 - content, 2, 98
 - in LIL, 28
 - synonymy, 98

- formal natural language semantics, 10
- formation tree (of a term), 9
- general locality condition, 46
- generalized variable, 20, 81
- generic
 - locality index, 42
 - locality proof schema, 70
- global meaning, 21
 - in LIL, 27
- head (of a term), 21
- immediate term, 20
- index, 11
- indexicality, 16
- intension, 11
 - Montague, 14
- intensional logic, IL, 10
- irreducible term, 21
- language of intensional logic, LIL, 12
- less than or equal to (locality index), 67
- level (of a type), 33
- local
 - associate (function), 33
 - preferred, 36
 - associate transformation, 82
 - closed locality index, 41
 - meaning, 24
 - in LIL, 27
 - synonymy, 24
 - term, 56
- locality, 2, 33
- locality condition, 35
 - general, 46
- locality index, 41
 - closed, 40
 - input, 40
 - local, 41
 - output, 40
 - standard, 41
 - constant, 40
 - generic, 42
 - less than or equal to, 67
 - more local than, 67
 - token, 41
 - variable, 42
- locality proof
 - closed, 58
 - standard, 66
 - more local than, 68, 71
 - most local, 69, 73
 - root, 60
 - schema, 69
 - generic, 70
- logic
 - intensional, IL, 10
 - language of intensional, LIL, 12
 - of demonstratives, LD, 16, 103
- meaning, 1
 - global, 21
 - in LIL, 27
 - local, 24
 - in LIL, 27
- metalanguage, 12
- more local than
 - locality index, 67
 - locality proof, 68, 71
- most local locality proof, 69, 73
- non local object, 34
- object-language, 12
- occurrence (of a variable)
 - bound, 9
 - free, 9
- operator
 - extension, 10
 - intension, 10

- part (of a term), 21
- preferred
 - associate (function), 48
 - local associate (function), 36
- proper term, 20
- pure type, 31
- pure variable, 18
- recursion, acyclic, 19
- recursive construct, 19
- recursive variable, 18
- recursor, 21
- reduction, 21
 - calculus, 22
 - rules, 21
- reference, 1
- referential
 - intension, 2, 20, 21
 - synonymy, 23
- render, 14
- root (of a locality proof), 60
- semantics, formal natural language, 10
- sense, 6
- sentence, 6
 - declarative, 6
- standard
 - closed locality index, 41
 - closed locality proof, 66
- state, 11
- state-dependent type, 32
- state-description, 11
- subproof, 60
- substitution, 42
- synonymy
 - factual, 98
 - local, 24
 - referential, 23
- term, 8
 - associate, 87
 - closed, 9
 - explicit, 23
 - factually synonymous, 98
 - formation tree of, 9
 - immediate, 20
 - irreducible, 21
 - local, 56
 - locally synonymous, 24
 - proper, 20
 - referentially synonymous, 23
- theory of referential intensions, 1, 18
- token, (locality) index, 41
- transformation
 - associate, 87
 - local associate, 82
- tree, formation (of a term), 9
- type, 8
 - associate, 44
 - basic, 8
 - level of, 33
 - pure, 31
 - state-dependent, 32
- universe, 9
- variable, 8
 - (locality) index, 42
 - generalized, 20, 81
 - occurrence
 - bound, 9
 - free, 9
 - pure, 18
 - recursive, 18