Chapter 4

Reasoning with Monotone Functions

In this chapter we present a proof theoretical account of natural reasoning using linguistic structures as the vehicle of inference. Following [Sup79, Ben86], we refer to this system as a Natural Logic. In particular, we focus attention on monotone inference. The notion of monotonicity is closely related to that of negative polarity items (NPIs). We aim to develop a system which, while marking the linguistic structures with the information required to model natural reasoning inferences, accounts for NPI distribution.

We start by showing the link between NPIs and natural reasoning (Section 4.1). While discussing this connection, we give the background behind the natural logic proposed by van Benthem [Ben86] and further studied by Sánchez [SV91] (Section 4.2). By looking at an alternative system proposed by Dowty in [Dow94], we show that a logical account of NPIs requires the use of internalized polarity markers (Section 4.3). Finally, in Section 4.4 we describe a natural logic based on a multimodal categorial type logic (MCTL). We show that MCTL has the required expressivity to account for NPIs and produce marked structures ready for deriving natural reasoning inferences.

4.1 Parsing and Reasoning

The task of accounting for the role of language in drawing inferences is commonly considered to belong to the domain of formal semantics. Most of the literature in natural reasoning assumes a model theoretic perspective, and uses a formal language as an intermediate step into which natural language expressions are translated. In this translation information is lost about natural language structures which might be relevant when drawing inferences. It is an interesting question whether a system can be developed which is able to compute natural reasoning inferences taking into consideration the information human beings rely on when reasoning. In particular, we are referring to the syntactic and semantic information obviously involved in reasoning. This is the task that the construction of a natural logic is supposed to tackle as explained in [Ben86, Ben87b] inspired by [Sup79], where the name indicates that instead of employing logical forms as vehicles of inference, natural language expressions are used directly.

In this approach, the applicability of an inference pattern is read off from a derivation. The advantage gained by assuming such proof theoretical perspective is that one can
Chapter 4. Reasoning with Monotone Functions

study how natural language structures contribute to natural reasoning. It is, of course, an ambitious project, given the high complexity of sentences, the richness of ambiguity which is a hallmark of any human language, and the variety of natural reasoning forms. In order to make it more tractable, we focus on specific inference patterns selected together with a restricted set of linguistic structures. Following [Ben86, SV91] we will look at monotonicity inferences. See [FWF00] for an interesting natural logic involving conservativity in addition to monotonicity.

Besides contributing to natural reasoning, monotonicity also plays an important role in establishing the grammaticality of some forms of linguistic composition. In natural language one finds expressions known as negative polarity items (NPIs), whose syntactic distribution is determined by the monotonicity property of the phrases they are in construction with. A system employed to model the composition of linguistic signs must be able to take monotonicity information into consideration when working with these expressions. In this chapter we show how the two aspects of parsing and reasoning can be accounted for within the categorial type logic framework. In particular, we present a system which can deal with the distribution of NPIs and produce marked parsed output from which monotone inferences can be drawn.

4.1.1 Negative Polarity Items and Monotonicity

The study of negative polarity items (NPIs) started with the work by Klima [Kli64] who looks at them as expressions which must be ‘in construction with’ a trigger or licensor, where the latter is either negation or an “affective element”, e.g. a verb like surprised. However, no explicit references to the existence of a phenomenon of negative polarity were made, yet. The move to a conceptualization of it and the introduction of the terminology now in use is due to Baker [Bak70]. In that work, the notion of licensing elements made an implicit appearance, since the distribution of polarity items is seen as a matter of being in the scope of a suitable element, though “scope” and “suitable element” are not properly defined yet. Fauconnier [Fau75] looked at NPIs as denoting extreme elements among a set of alternatives. In doing so, he introduced a semantic perspective on the issue. This work inspired Ladusaw [Lad79], who gave a precise semantic interpretation to the vague idea of affective licensors proposed by Klima, identifying them with downward monotone expressions. Since then, negative polarity items have been studied from various perspectives, some more syntactically oriented [Lin81, Pro88], others more semantic in nature [Zwa86, Wou94, Gia97] and yet others more focused on the pragmatic aspects [KL89, KL93, Chi02]. In Chapter 7, we analyze the distributional behavior of NPIs in detail, but for now it is sufficient to define them as follows.

\textbf{Definition 4.1.} [Negative Polarity Items] Negative polarity items are expressions which can appear felicitously only in the scope of monotone decreasing functions.

This definition calls for another one used to identify NPIs’ licensors.

\textbf{Definition 4.2.} [Monotone Functions] Let \( f : A \rightarrow B \) be a function and let \( \leq_A, \leq_B \) be partial orders on \( A \) and \( B \), respectively. Then,
4.1. Parsing and Reasoning

a. $f$ is monotone increasing ($\uparrow\text{Mon}$) iff $\forall x, y \in A, x \leq_A y$ implies $f(x) \leq_B f(y)$.

b. $f$ is monotone decreasing ($\downarrow\text{Mon}$) iff $\forall x, y \in A x \leq_A y$ implies $f(y) \leq_B f(x)$.

The $\uparrow\text{Mon}$ functions are also referred to as upward monotone and the $\downarrow\text{Mon}$ functions as downward monotone. Let us illustrate the meaning of these definitions when applied to linguistic data.

In the classical categorial grammar (CG) approach to natural language, the derivation of sentences is seen as a sequence of function applications starting from the categories assigned to the lexical items in the lexicon. The original references [Ajd35, BH53] deal mainly with syntax, and treat composition as type combination. However, the framework lends itself so naturally to the formal analysis of monotonicity phenomena, that the categorial grammar literature about the application of the monotonicity calculus to natural language is quite broad. A classic reference is [Zwa86] where these kinds of semantic properties are investigated from an algebraic perspective. This methodology permits very sharp semantic analyses of monotonicity phenomena, such as generalized quantifiers. Let us first recall the definition of partially ordered domains we discussed in Section 1.3, which can be used to check the monotonicity property of linguistic signs.

**Definition 4.3.** [Partially Ordered Domain] Let $Dom_a$ be a domain of type $a$, where $a \in \text{TYPE}$ and $\text{TYPE}$ is built over the set $\{e, t\}$.

- If $\beta, \gamma \in Dom_e$, then $[\beta] \leq_e [\gamma]$ iff $[\beta] = [\gamma]$.
- If $\beta, \gamma \in Dom_t$, then $[\beta] \leq_t [\gamma]$ iff $[\beta] = 0$ or $[\gamma] = 1$.
- If $\beta, \gamma \in Dom_{(a, b)}$, then $[\beta] \leq_{(a, b)} [\gamma]$ iff $\forall \alpha \in Dom_a, [\beta(\alpha)] \leq_b [\gamma(\alpha)]$.

Once we know the monotonicity of linguistic expressions, we also know how to deal with NPIs. For instance, Definition 4.1 correctly predicts the data below where the NPI $yet$ and $anybody$ are licensed by the $\downarrow\text{Mon}$ function $\text{nothing}$ (1-a) and $\text{doubt}$ (1-c) and are ungrammatical in the scope of the $\uparrow\text{Mon}$ function $\text{everybody}$ (1-b) and $\text{think}$ (1-d).

(1)  
  a. *Nobody left yet.
  b. *Everybody left yet.
  c. John doubts that anybody left.
  d. *John thinks that anybody left.

As soon as we move to consider more complex sentences, an important information about monotone functions, is the way they compose. For example, in (2-a) $anybody$ is ungrammatical since $didn’t$ and $\text{doubt}$ compose yielding an upward monotone function [Tov96]. Similarly, in (2-b) the two downward monotone functions $\text{not}$ and $\text{all}$ compose together and therefore cannot license $\text{anything}$ [Hoe86].

(2)  
  a. *John didn’t doubt that anybody left.
  b. *Not all students who know anything about logic know Modus Ponens.

Formally, monotone functions compose as follows.

**Proposition 4.4.** [Monotone Functions Composition] It is straightforward to prove that the composition of monotone functions follows a sign rule. Let $A^+ \rightarrow B$ ($A^- \rightarrow B$) stand
for an upward (resp. downward) monotone function \( f : A \to B \). For \( g : A^x \to B \) and \( f : B^y \to C \), \( f \circ g : A^{sg(x,y)} \to C \), where \( sg(x,y) = + \) for \( x = y \), and \( - \) otherwise. Expressed as a table:

\[
\begin{array}{|c|c|}
\hline
f \circ g & h \\hline
\uparrow \text{Mon} \circ \uparrow \text{Mon} = & \uparrow \text{Mon} \\hline
\downarrow \text{Mon} \circ \downarrow \text{Mon} = & \uparrow \text{Mon} \\hline
\uparrow \text{Mon} \circ \downarrow \text{Mon} = & \downarrow \text{Mon} \\hline
\downarrow \text{Mon} \circ \uparrow \text{Mon} = & \downarrow \text{Mon} \\hline
\end{array}
\]

Finally, Linebarger [Lin81] shows that negative polarity items must occur in the immediate scope of their licensor with no logical elements intervening, where logical elements are expressions able to entering into scope ambiguities. For instance, in (3-a) the reading with every as an intervener between the negation and the negative polarity item is ungrammatical. However, there also exist harmless interveners like the bridge predicate think which does not block the link between the NPI and its licensor (3-c). Moreover, multiple NPIs can be licensed by the same trigger as in (3-d) taken from [Lad92]. Let \([X > Y]\) mean ‘X has scope over Y’, and let % mark awkward sentences.

(3) a. Mary didn’t wear any earrings to every party. [Neg > Any > Every].  
   b. %Mary didn’t shout that John had any problems. 
   c. Mary didn’t think that John had any problems.  
   d. Nobody said anything to anybody.

The contrast between (3-b) and (3-c) has been explained by assuming that non-bridge verbs like shout are essentially quotational and hence embed a structure that contains an illocutionary operator [Kri95]. These examples show that in addition to the logical composition of monotone functions, what matters is the way in which linguistic signs are actually assembled. Therefore, a deductive account of negative polarity items must allow for controls on the way functions compose. We will come back to this point in Sections 4.3 and 4.4. First, let us look at the role monotone functions play in drawing inferences.

### 4.1.2 Monotonicity in Natural Reasoning

Looking at the definition of monotone functions, it can be seen that monotonicity is closely tied to natural reasoning, since the partial order \( \leq \) (Definition 4.3) can be interpreted as the entailment relation. Let us clarify this connection by means of an example.

**Example 4.5.** [Natural Reasoning Patterns] Consider these intuitively correct inferences:
4.1. Parsing and Reasoning

Everybody (left something expensive)  (A1)  Nobody (left yet)  (B1)
Everybody (left something)  (A1)  Nobody (left in a hurry yet)  (B1)
Some boy will run fast  (A2)  No boy will run  (B2)
Some boy will run  (A2)  No boy will run fast  (B2)
Not every good logician wanders  (A3)  Every logician wanders  (B3)
Not every logician wanders  (A3)  Every good logician wanders  (B3)

These inferences are valid only because something is “more general” than something expensive; left is “more general” than left in a hurry; and run and logician are more general than run fast and good logician, respectively. Formally, they involve replacing one expression by another, the denotation of which is a superset (as in A) or a subset (as in B) of the denotation of the original expression.

We aim to derive the above inferences proof theoretically from parsed marked sentences. In order to achieve this goal, first of all we need to have a clearer picture of the kind of inferences we need to model. A substitution of an expression with something more or less general could in fact be done in any position in a sentence. Therefore, the described behavior can be expressed in more general terms by the following inference schemas. Let $P$ and $Q$ stand for linguistic expressions, and let $\leq$ be a partial order between their denotations, respecting their degree of “generality” (see Example 1.31). Then, if $[P] \leq [Q]$,

$$\frac{N[P]}{N[Q]} \text{ (A) or } \frac{N[Q]}{N[P]} \text{ (B)}$$

The examples (A1), (A2) and (A3) instantiate (A), while (B1), (B2) and (B3) exemplify (B). The question which arises at this point is whether these schemas can always be applied, and how we can decide which of the two produces a valid inference. An answer to this question is given by the monotonicity calculus, and its connection to the computation of the polarity of the position in which the expression occurs.

Before looking at the relation between monotonicity and polarity in formal languages, we want to draw attention on the fact that there exist also environments which do not allow inferences in any direction [Hoe86]. These contexts are created by nonmonotone functions like exactly three boys. For example, from exactly three boys were skating, one cannot derive either exactly three boys were moving nor exactly three boys were skating fast. There are two sorts of nonmonotone functions: those which block any kind of substitution, also known as opaque or intentional contexts like believe, and those which still allow substitution of equivalent expressions. In the remainder of the chapter we will not take the nonmonotone functions into consideration, but the systems described here could be extended to include them as well.

4.1.3 Monotonicity and Polarity

Monotonicity and polarity are well-known notions in mathematics [Lyn59]. We have already anticipated the definition of monotone functions, what we need to make explicit is the flow of information from the function to its argument.
Chapter 4. Reasoning with Monotone Functions

Definition 4.6. [Monotone Argument Positions] The argument of an ↑Mon function \( f : A \rightarrow B \) is said to be in an increasing monotone position, whereas the argument of a ↓Mon function is in a decreasing monotone position.

When interpreting linguistic expressions as functions and the construction of sentences as function composition, the monotonicity calculus gives us the required information to decide which one of the inference schemas given above can be applied for drawing correct inferences: an expression in an increasing (resp. decreasing) monotone position is substituted with a more (resp. less) general one following (A) (resp. (B)). Let us go back to Example 4.5 and consider again the inference in (B2), trying to apply the intuitions we just discussed.

Example 4.7. Suppose we take a functional perspective and consider the linguistic phrases no boy and will as functions represented as no\_boy and will and the sentence No boy will run as the result of the functional application of the composed function no\_boy \( \circ \) will to run. Using the standard linguistic categories, the function representing run is of category iv (intransitive verb), and the whole sentence is s (sentence).

\[
\begin{align*}
\text{no\_boy} : & \text{iv}^- \rightarrow s \\
\text{will} : & \text{iv}^+ \rightarrow \text{iv} \\
\text{no\_boy} \circ \text{will} : & \text{iv}^{sg(+-)} \rightarrow s \\
\text{no\_boy} \circ \text{will} : & \text{iv}^- \rightarrow s \\
\text{no\_boy} \circ \text{will} \circ \text{run} : & s \\
\end{align*}
\]

(1) no boy will run

\[
\begin{align*}
\text{no\_boy} \circ \text{will} & : \text{iv}^- \rightarrow s \\
\text{no\_boy} \circ \text{will} & : \text{iv}^- \rightarrow s \\
\text{no\_boy} \circ \text{will} \circ \text{run} & : s \\
\end{align*}
\]

The monotonicity of the composed function no\_boy \( \circ \) will gives us the information required to derive the inference above. But suppose we want to derive no boy or no girl will run from (1). The monotonicity calculus does not help us there, as we are substituting in a function position, instead of in an argument position. We need to extend the monotonicity calculus, so that it will also yield marked positions for these cases, and this is not as straightforward as it seems, because we need a way to refer explicitly to the functions. We need to compute the polarity of the position where the function occurs.

In first order logic (FOL) polarity is defined in terms of the number of negations surrounding a subformula [Lyn59]. But FOL does not capture the compositional behavior needed for formalizing natural language functional application. In Section 1.3, we have seen that a way of properly speaking about functions is provided by the lambda calculus [Chu40]. By introducing it into the picture we have the final missing ingredient.

Briefly, the set of typed lambda terms is defined as follows. For any type \( a \), let \( \text{CON}_a \) be a set of constants of type \( a \) and \( \text{VAR}_a \) be a set of variables of type \( a \), then the set of typed lambda \( \text{TERM}_a \) expressions over \( \text{CON}_a \) and \( \text{VAR}_a \) is given by

\[
\begin{align*}
\text{TERM}_{a} & := \ c_a \ | \ x_a \ | \ (\text{TERM}_{(b,a)} \ \text{TERM}_{b}) \\
\text{TERM}_{(b,a)} & := \ \lambda x_b.\text{TERM}_{a}.
\end{align*}
\]
where \( c_a \in \text{CON}_a \) and \( x_a \in \text{VAR}_a \). The operator \( \lambda \) is said to *bind* \( x \); unbounded variables are said to be *free*, we abbreviate with \( FV(M) \) the set of free variables in \( M \). For this language it is now possible to define both monotone and polarity positions. The definitions were originally given in \([\text{Ben86}]\) and further explored in \([\text{SV91}]\).

**Definition 4.8.** [Monotone Occurrence] Let \( N'_a \) be a lambda term like \( N_a \) except for containing an occurrence of \( M'_b \) where \( N_a \) contains \( M_b \) (viz. \( N'_a = N_a[M'_b/M_b] \)), where \( a, b \) stand for the type of the indexed terms.

1. \( N_a \) is *upward monotone in* \( M_b \) iff for all models \( \mathcal{M} \) and assignments \( f: \lbrack M \rbrack_{\mathcal{M}} \leq_b \lbrack M' \rbrack_{\mathcal{M}} \Rightarrow \lbrack N \rbrack_{\mathcal{M}} \leq_a \lbrack N' \rbrack_{\mathcal{M}} \);\)
2. \( N_a \) is *downward monotone in* \( M_b \) iff for all models \( \mathcal{M} \) and assignments \( f: \lbrack M \rbrack_{\mathcal{M}} \leq_b \lbrack M' \rbrack_{\mathcal{M}} \Rightarrow \lbrack N \rbrack_{\mathcal{M}} \leq_a \lbrack N' \rbrack_{\mathcal{M}} \).

**Definition 4.9.** [Polarity of Occurrences] Given a lambda term \( N \) and a subterm \( M \) of \( N \). A specified occurrence of \( M \) in \( N \), is called *positive* (negative) according to the following clauses:

1. \( M \) is positive in \( M \).
2. \( M \) is positive (negative) in \( PQ \) if \( M \) is positive (negative) in \( P \).
3. \( M \) is positive (negative) in \( PQ \) if \( M \) is positive (negative) in \( Q \), and \( P \) denotes an upward monotone function.
4. \( M \) is negative (positive) in \( PQ \) if \( M \) is positive (negative) in \( Q \), and \( P \) denotes a downward monotone function.
5. \( M \) is positive (negative) in \( \lambda X.P \) if \( M \) is positive (negative) in \( P \) and \( X \not\in FV(M) \).

Having added the lambda notation we can complete Example 4.7. The sentence *no boy will run* can be represented by \((\text{no boy will})\text{run}\), where \( \text{no boy}, \text{will} \) and \( \text{run} \) are constants: the first one is a downward monotone function and the others two are upward monotone. By Definition 4.9, the function \((\text{no boy})\) receives a positive polarity, and given than *no boy or no girl* is more general that *no boy* we are allowed to monotonically conclude *no boy or no girl will run* from *no boy will run*.

The relation between monotonicity and polarity in lambda terms has been studied in \([\text{Ben86}, \text{Ben91}, \text{SV91}]\) where it is proved that the polarity of the occurrences implies their monotonicity.

**Proposition 4.10.** If \( M_b \) is positive (resp. negative) in \( N_a \), then \( N_a \) is upward (resp. downward) monotone in \( M_b \).

**Corollary 4.11.** If \( X_a \) is positive (resp. negative) in \( N_b \), then \( \lambda X_a.N_b \) denotes an upward (resp. downward) monotone function.

The differences between monotonicity and polarity could be summarized in a few words by saying that monotonicity is a semantic property of functions which is dynamically passed to the argument positions while building a formula. On the other hand, polarity is a static syntactic notion which can be computed for all positions in a given formula. This connection between the semantic notion of monotonicity and the syntactic one of
polarity is what one needs to reach a proof theoretical account of natural reasoning and build a natural logic.

Categorial type logic (CTL) derivations can be interpreted as lambda terms due to the Curry-Howard correspondence [CF68, How80, Ben88] (see Section 1.4). This correspondence makes it possible to associate CTL derivations with the notion of polarity on lambda terms (Definition 4.9) and consequently, by Proposition 4.10, with their monotone positions [Ben86]. In other words, CTL derivations can supply the information required for deriving monotone inference from the parsed linguistic structures.

By linking up the two topics here introduced, we recall that our aim is twofold. We want to encode monotonicity information and compute the polarity positions within parsed structures so as to account for negative polarity distribution and produce structures readily available for deriving monotone inferences.

### 4.2 A Natural Logic based on LP

In [SV91] Sánchez works out the proposal of van Benthem [Ben86]. The syntactic issues of function composition is connected to the semantic features of monotonicity inference. This link is obtained extra-logically by means of an algorithm working on derivations of the associative and commutative Lambek calculus (LP) (Section 1.2), which is able to correctly mark the parsed strings determining the different polarity positions. Due to the use of an external algorithm to compute the polarity of the nodes in LP derivations, we will refer to this system as LP+£Pol. For the ease of presentation here we use our notation while describing LP+£Pol derivations. As a consequence we use two implicational operators \( \land, \lor \) though in LP they collapse into one. The original presentation is given in Appendix A, where LP+£Pol is embedded into a multimodal categorial type logic.

Given a derivation, the algorithm starts from marked leaves and propagates these markers through the proof by labelling the nodes of the logical rules. First, the logical types in each leave are enriched with monotonicity markers encoding the monotonicity property of the corresponding linguistic entry. Let \( A/B \) be the type assigned to an upward (resp. downward) monotone function, then \( A/B^+ \) (resp. \( A/B^- \)) is the marked type. Similarly for \( B\setminus A \). Types corresponding to variables in the lambda terms are left unspecified \( A/B, (A\setminus B) \). From this information, the algorithm proceeds by propagating the monotonicity markers +, − from the leaves through the derivation. Finally, the polarity of the nodes in the derivation is computed: a negative marker flips the monotonicity of all nodes above it in the derivation, \( \text{viz.} + \) becomes − and \( \text{vice versa} \), as required by Proposition 4.4 and Definition 4.9. An unmarked node breaks the polarity assignment leaving the nodes above it in the branch unspecified. The final result is a parsed output in which polarity positions are correctly displayed, and which can be used as a vehicle of natural reasoning inference. The formal definition of the algorithm consists of the two parts below.

**Monotonicity Markers:** Let \( \mathcal{D} \) be a derivation of LP, and \( \mathcal{D}' \) be the derivation \( \mathcal{D} \) with the types of the lexical items marked according to their monotonicity properties. The monotonicity markers are copied from the functions to their arguments in the derivation.
by means of the rewrite rules in Figure 4.1 where $x \in \{+,-\}$. Similar rules hold for $\setminus$. The value of $y$ in $[/I]^i$ is determined as follows: $y$ is $-$ (resp. $+$) if all the nodes in the path from $\Gamma \vdash A \setminus B$ to $[A \vdash A]^i$ are marked, and the number of nodes marked with $-$ in the path is odd (resp. even). As a last step, the monotonicity algorithm marks the root of the derivation with $+$. 

**Polarity Markers:** Given a marked derivation $\mathcal{D}$ a node is assigned a polarity marker $+$ or $-$ if all the nodes in the path from the node to the root are marked. The node is $-$ if the number of nodes marked with $-$ in the path is odd, and $+$ otherwise.

Note that unspecified polarity nodes will occur only in derivations involving abstraction. They are in fact produced when the formula corresponding to the node is taken as argument by variables, or in other words when it is the minor premise of a functional application $[/E]$ or $[\setminus E]$ having a hypothesis (or a subformula of a hypothesis) in the major premise. Let us see this algorithm at work.

**Example 4.12. [LP+EPol Functional Applications]** Given the set of lexical entries \{not $\in$ s/s, wanders $\in$ np/s, good logician $\in$ n, every $\in$ (s/(np\s))/n\}, the sentence *not every good logician wanders* is proved to be of type $s$ in LP as follows:

$$
\frac{\text{every } \vdash (s/(np\s))/n \quad \text{good logician } \vdash n}{\text{not } \vdash s/s} \quad \frac{\text{every } \circ \text{good logician } \vdash s/(np\s)}{\text{wanders } \vdash np\s} \quad [/E] \\
\frac{\text{not } \circ ((\text{every } \circ \text{good logician}) \circ \text{wanders} ) \vdash s}{[/E]}
$$

Now, we need to encode the monotonicity information for the lexical entries, namely the leaves of the derivation. From formal semantics, we know that *not* and *wanders* are downward and upward monotone functions, respectively; whereas *every* is downward monotone in its first argument and upward monotone in its second argument. The derivation is labelled accordingly by replacing the functional types at the leaves of the above derivation with the corresponding marked ones: $s/s^-, (s/(np\s)^+)/n^-$ and
\(np^+\backslash s\). These monotonicity markers are then propagated through the derivation as illustrated below in (a). Finally, the polarity positions of the nodes are computed from the monotone ones in (b). We give below the skeleton of the marked derivations.

(a) **Monotonicity markers**

\[
\begin{array}{ccc}
(s/(np\backslash s)^+)/n^- & n & s/(np\backslash s)^+ \\
+ & - & + \\
\hline
s/(np\backslash s)^+ & np^\backslash s & s/s^- \\
+ & + & - \\
\hline
s & s & + \\
+ & - & +
\end{array}
\]

(b) **Polarity markers**

\[
\begin{array}{ccc}
(s/(np\backslash s))/n^- & n & s/(np\backslash s) & np\backslash s \\
- & + & - & + \\
\hline
s/(np\backslash s) & np\backslash s & s/s & s \\
- & + & - & - \\
\hline
s & s & + & +
\end{array}
\]

Note how in (a) the negative marker carried by ‘not’, being a downward monotone function, is passed to the expression taken as argument, ‘every good logician wanders’. This reflects what was expressed in Definition 4.6. Moreover, in (b) the negative monotonicity marker assigned to the corresponding node flips (reverses) the markers assigned to the nodes above it, mirroring Definition 4.9-(iv). For each node in the polarity marked derivation, if a node is labelled with a + (−), the corresponding term in the final lambda term representing the whole sentence, \(\text{Not}((\text{Every good logician}) \text{ wanders})\), is in a positive (negative) position. For instance, the node \(n\) is assigned a + and the lambda term corresponding to it, \(\text{good logician}\), is in a positive polarity position. Finally, note that the root of the derivation receives a positive polarity marker, resembling the first point of Definition 4.9, \(M\) is positive in \(M\).

We can now display the polarity markers on the linguistic structures corresponding to each node obtaining a marked parsed string: \((\text{not}^+((\text{every}^- \text{ good logician}^+)^- \text{ wanders}^-)\)^+. Due to the link between polarity and monotonicity (Proposition 4.10 and Definition 4.8), this structure can be used to derive monotone natural reasoning inferences. For instance, the inference (A3) in Example 4.5 can be derived by replacing ‘good logician’ with the more general term ‘logician’. We use the thick inference line to distinguish this new inference from the logical and structural rules of the logical system.

\[
\text{not}^+((\text{every}^- \text{ good logician}^+)^- \text{ wanders}^-)\) + \vdash s
\]

Note that the information regarding the partial order holding among the expression involved in the substitution is still based on formal semantics and it is computed on the corresponding lambda terms. See [FWF00] for a natural logic where the order relation is computed within the system.

Besides functional applications, LP derivations can contain abstraction rules. As an example of the latter, we look at the lifting of an np to a higher order type.

**Example 4.13.** [Abstraction in LP+EPol] Let mary \(\in np\) be our lexicon entry. The lifted type \(s/(np\backslash s)\) is obtained as follows:
4.2. A Natural Logic based on LP

\[
\frac{\text{mary } \vdash \text{np } [x \vdash \text{np}s]^1}{\text{mary } \vdash \text{s}/(\text{np}s)} \quad [\text{E}]^1
\]

Following the algorithm, since \( x \) is a variable, it is an unspecified monotone function and its type is left unmarked.

(a) **Monotonicity markers**

\[
\frac{np \quad [np\backslash s]^1}{s} \quad [\text{E}]^1
\]

(b) **Polarity markers**

\[
\frac{s \quad s}{s/(np\backslash s)+} \quad [\text{I}]^1
\]

The point of interest in this case is the role played by the assumed function \( x \). Since its monotonicity could be either positive or negative, the monotone argument position where ‘mary’ occurs is also unspecified. Furthermore, when the hypothesis is discharged an upward monotone function is built: the abstraction is over a variable in a positive polarity position (Corollary 4.11).

The role of the hypothetical reasoning in the assignment of polarity positions can be better illustrated by looking at an example involving coordination of an \( np \) with a higher order type, requiring the application of the lifting theorem above.

**EXAMPLE 4.14.** [Coordination] The coordination \( \text{and} \) is an upward monotone function in both arguments. Its lambda term representation is: \( \lambda RQZ.QZ \land RZ \). The lambda term of the coordinated phrase is built as follows.

\[
\text{m} \vdash m \quad \text{and} \vdash \lambda RQZ.QZ \land RZ \quad \text{every}\_\text{logician} \vdash \text{Every}\_\text{logician}
\]

\[
\text{mary} \vdash \lambda P.Pm \quad (\text{and} \circ \text{every}\_\text{logician}) \vdash \lambda QZ.QZ \land \text{Every}\_\text{logician} Z
\]

\[
\text{mary} \circ (\text{and} \circ \text{every}\_\text{logician}) \vdash \lambda Z.Zm \land \text{Every}\_\text{logician} Z
\]

In the final lambda term the polarity of ‘mary’, represented by \( m \), is still unspecified. It depends on the monotonicity of the verb phrase which will be taken as an argument and will replace the variable \( Z \). For instance, if the coordinated structure is applied to the verb phrase \( \text{left} \) it yields \( \text{mary and every logician left} \) represented by: \( \text{Left} \text{m} \land \text{Every}\_\text{logician} \text{Left} \) where \( m \) is in a positive polarity position.

**Monotonicity Marking**

\[
\text{mary} \vdash np \quad \ldots \quad \text{and} \vdash ((s/(np\backslash s))^+/(s/(np\backslash s))^+)/s/(np\backslash s))^+ \quad \text{every}\_\text{logician} \vdash s/(np\backslash s)^+
\]

\[
\text{mary} \vdash s/(np\backslash s)^+ \quad (\text{and} \circ \text{every}\_\text{logician}) \vdash (s/(np\backslash s))^+/(s/(np\backslash s))^+
\]

\[
(mary \circ \text{and}) \circ \text{every}\_\text{logician} \vdash s/(np\backslash s)^+
\]
Sánchez’ algorithm does not assign a polarity to ‘mary’, leaving it unspecified even after the application of the coordinated structure to a verb phrase, since the assignment of the polarity requires that all the nodes must be marked and the leaf ‘mary † np’ is not.

By extending the result in [Ben91] Sánchez proves in [SV91] the soundness of \( \text{LP+EPol} \). By interpreting derivations as lambda terms, marked functional nodes will correctly correspond to monotone functions, and polarity markers will correctly correspond to polarity positions in the lambda terms\(^1\). Therefore, \( \text{LP+EPol} \) properly accounts for the tasks it was built for: it generates marked parsed outputs ready for deriving monotonicity inference. But this was not our original aim. Let us look back at what we wanted to achieve.

Our project was to build a system able to (i) encode monotonicity information, (ii) compute polarity positions, (iii) use this information while parsing (to verify correct use of NPIs) and (iv) produce marked output from which monotonicity inferences can be derived. Though \( \text{LP+EPol} \) can account for (i),(ii) and (iv), it does not account for (iii). This shortcoming is due to the fact that the monotonicity and polarity algorithms are external to the logic: they mark the derivation only after the grammaticality of the linguistic structure is established. Hence, polarity markers do not play an active role in the derivation and cannot be used to control the grammaticality of linguistic structures as required by the NPIs. A solution can be obtained by internalizing the marking algorithms.

### 4.3 Internalizing Polarity Marking in CG

Dowty [Dow94] proposes a natural logic based on classical categorial grammar (CG+Pol), in which the independent steps of monotonicity and polarity marking collapse into a syntactic derivation. In this approach the markers ‘+’ and ‘-’ are used to indicate the final polarity. The main characteristics of Dowty’s system are:

a. Since one and the same word can appear with positive polarity in one derivation and with negative polarity in another, most lexical items—with the important exception of negative polarity items\(^2\)—will have both a ‘+’ and a ‘-’ marked category, with the same interpretation.

b. \( \uparrow \text{Mon} \) functions are assigned categories of the forms \( A^+ / B^+ \) and \( A^- / B^- \), meaning that they preserve the polarity markers. We will mark them as \( A^x / B^x \).

c. \( \downarrow \text{Mon} \) functions are assigned categories of the forms \( A^+ / B^- \) and \( A^- / B^+ \), since they reverse the polarity markers. We will mark them as \( A^x / B^y \).

For complex categories we use the convention: \( (A/B)^x =_{df} (A^x/B)^x =_{df} (A^x/B) \), and similarly for \((A \backslash B)^x\)^3.

\(^1\)The case discussed in the Example 4.14 does not constitute a problem for the soundness proof which only concerns marked nodes.

\(^2\)Dowty considers also positive polarity items, as expressions required to occur in a positive polarity context.

\(^3\)This definition is based on the fact that in a function-argument combination the function always has the same polarity as the combination as a whole.
Functional application respects the polarity markers in the following way. Let \( x, y \in \{+, -\} \), then

\[
A^x/B^y \quad B^y \\
\hline
A^x
\]

where ‘\( x \)’ and ‘\( y \)’ coincide when the major premise \( A^x/B^y \) is an \( \uparrow \)Mon function, and differ when it is a \( \downarrow \)Mon function.

The grammar thus defined generates sentences of category \( S^+ \) or \( S^- \). The former is the category of independent (grammatical) sentences, the latter of sentences embedded inside a \( \downarrow \)Mon function. We will assume the lexical entries below. Let \( x \in \{+, -\} \), let \( y \) be the “opposite” of \( x \), and let \( VP = NP\backslash S \),

**Lexical Entries**

<table>
<thead>
<tr>
<th>( \text{walks} )</th>
<th>( \text{reads} )</th>
<th>( \text{John} )</th>
<th>( \text{doesn’t} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( VP^x )</td>
<td>( VP^x/VP^x )</td>
<td>( S^x/VP^x )</td>
<td>( VP^x/VP^y )</td>
</tr>
</tbody>
</table>

We present some examples of Dowty-style polarity marking.

**EXAMPLE 4.15. [Polarity Marking]**

1. *John walks.*

\[
\begin{array}{c|c}
\text{John} & \text{walks} \\
\hline
S^+/VP^+ & VP^+ \\
\end{array}
\]

2. *John doesn’t walk.*

\[
\begin{array}{c|c|c}
\text{John} & \text{doesn’t} & \text{walk} \\
\hline
S^+/VP^+ & VP^+/VP^- & VP^- \\
\end{array}
\]

Comparing these two derivations, note how the polarity of the \( VP \) is changed by the presence of the \( \downarrow \)Mon function ‘doesn’t’. Using a bottom-up reading, the inference says that if ‘John doesn’t walk’ is a well-formed independent sentence \( S^+ \), then ‘John’, the last function applied, has to have \( S^+ \) as a value. Since ‘John’ is an \( \uparrow \)Mon function, this means its argument must be marked with a ‘+’ as well. This requires ‘doesn’t walk’ to be of category \( VP^+ \). Therefore, the value of the \( \downarrow \)Mon function ‘doesn’t’ must be marked with the ‘+’ as well, and consequently its argument is marked negatively.

To deal with more complex sentences in which a generalized quantifier occurs in the object position, Dowty includes in the lexicon for each determiner \( (S^x/VP^y)/CN^z \) an object counterpart, of category \( (TV^y\backslash VP^x)/CN^z \), where \( TV^y = (NP^y\backslash S^y)/NP^y \). The object counterparts of the above given lexical entries ‘a’, ‘no’ and ‘any’ are as follows:
\[ a = (TV^x \backslash VP^x) / CN^x \]
\[ \text{any} = (TV^- \backslash VP^-) / CN^- \]
\[ \text{no} = (TV^x \backslash VP^y) / CN^x \]
\[ \text{every} = (TV^x \backslash VP^x) / CN^y \]

**Example 4.16.** [Negative Polarity Item]

1. *No boy reads any book.*

\[
\begin{array}{c|c|c|c|c}
\text{no} & \text{boy} & \text{reads} & \text{any} & \text{book} \\
(S^+ / VP^-) / CN^- & CN^- & TV^- & TV^- \backslash VP^- & CN^- \\
S^+ / VP^- & & VP^- & & \\
S^+ & & & & \\
\end{array}
\]

If we replace ‘no’ with a determiner which is upward monotone in its second argument, e.g. ‘every’, the derivation fails. In order to match the \( VP^- \) category of \( \text{reads any book} \), ‘every’ has to be considered of category \( (S^- / VP^-) / CN^+ \) and the whole expression \( \text{every boy reads any book} \) would be proved to be of category \( S^- \) rather than \( S^+ \), where \( S^- \) is the category of the embedded sentence in the scope of the \( \downarrow \text{Mon} \) function.

This example brings us to look at NPIs in embedded sentences. Remember from Section 4.1 that NPIs in embedded sentences may be licensed by a downward monotone function in the matrix clause in two cases: either they are the object of a negative predicate (1-c), or there is a predicate bridging the NPI to its licensor (3-c). On the other hand, the composition of a negative predicate with another downward monotone function cancels their licensing property by yielding an upward monotone function (2-a), and an intervener blocks the licensing relation (3-a).

The polarity marking of \( \text{CG} + \text{Pol} \) correctly predicts (1-c) and (2-a) as shown by the example below, where the lexical entry for \( \text{doubt} \) is \( VP^y / S^x \) (as it is a \( \downarrow \text{Mon} \) function).

**Example 4.17.** [Embedded NPIs]

1. *John doubts anybody left.*

\[
\begin{array}{c|c|c|c|c}
\text{John} & \text{doubts} & \text{anybody} & \text{left} \\
S^+ / VP^+ & S^- / VP^- & S^- & VP^- \\
S^+ & VP^- & & \\
\end{array}
\]

2. *John didn’t doubt anybody left.*

\[
\begin{array}{c|c|c|c|c}
\text{John} & \text{didn’t} & \text{doubts} & \text{anybody} & \text{left} \\
S^- / VP^- & VP^- / VP^+ & VP^+ & VP^- \\
S^- & & & \\
\end{array}
\]

However, the marking does not correctly account for the sentence in (4) since \( \text{every boy reads any book} \) is assigned category \( S^- \) which can be taken as argument by \( \text{doubt} \) contrary to linguistic reality: the quantifier \( \text{every boy} \) works as an intervener blocking the licensing of the NPI.
4.4 Internalizing Monotonicity and Polarity Markers in MCTL

In this section, we present a natural logic based on a multimodal categorial type logic with (polarity) structural rules (MCTL+Pol) where the latter are used simply as a tool for computing polarity position. Since the polarity of any position is positive unless modified by downward monotone functions, we leave the upward monotone functions unmarked and employ the unary operator $\ominus$ to mark downward monotone functions $\mathbf{4}$. Consequently, the corresponding unary structural connective $\ominus^\mathbf{4}$ marks a structure in a downward monotone argument position. Due to the logical relation holding between $\ominus$ and $\boxplus^\mathbf{4}$ (Section 2.1.2), and the link between monotonicity and polarity (Section 4.1.3), $\boxplus^\mathbf{4}$ encodes (negative) polarity information, as we will show below.

The dynamic flow of information from the function to the argument (Section 4.1.3) is directly accounted for by the logical rules without the need of an external monotonicity marking algorithm. Similarly, structural rules allow us to internalize the polarity algorithm producing marked structures instead of marking the corresponding nodes. A first clear advantage of the move from LP to MCTL is that the marked structures are readily available for deriving monotone inference, improving on the natural logic based on LP+EPol where the polarity marker were ‘externally’ displayed on the structures once read off the nodes. Furthermore, NPI distribution can be controlled, since polarity information is encoded in the logical types.

For the sake of simplicity we consider a product-free logic, similar to the systems considered so far. We repeat below the logical and structural languages of MCTL.

---

4 Note that the mode index is simply used to emphasize its role as a downward monotone marker.
full system is presented in Chapter 2, and an introduction of the linguistic application of the binary and unary operators is given in Sections 1.2 and 3.1.

**Definition 4.18.** [Languages] Logical Language: Given a set of basic categories $\text{ATOM}$, the set of categories $\text{FORM}$ is built over $\setminus, /, \otimes$ and $\sqsubseteq^\dagger$

$$\text{FORM} := \text{ATOM} \mid \text{FORM} / \text{FORM} \mid \text{FORM} \setminus \text{FORM} \mid \otimes \text{FORM} \mid \sqsubseteq^\dagger \text{FORM}.$$ 

Structural Language: The set of structures $\text{STRUCT}$ is built over the set of logical categories, by means of $\circ$, and $\langle \cdot \rangle^−$.

$$\text{STRUCT} := \text{FORM} \mid \langle \text{STRUCT} \rangle^− \mid \text{STRUCT} \circ \text{STRUCT}.$$ 

These languages are used to encode monotonicity and polarity information as summarized in Table 4.1.

<table>
<thead>
<tr>
<th>To express that:</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>A structure $\Gamma$ is an $\uparrow$Mon function</td>
<td>$\Gamma \vdash B/A$ or $\Gamma \vdash A\setminus B$</td>
</tr>
<tr>
<td>A structure $\Gamma$ is a $\downarrow$Mon function</td>
<td>$\Gamma \vdash B/\otimes A$ or $\Gamma \vdash \otimes A\setminus B$</td>
</tr>
<tr>
<td>A structure $\Gamma$ has polarity $−$</td>
<td>$\langle \Gamma \rangle^−$</td>
</tr>
<tr>
<td>A structure $\Gamma$ has polarity $+$</td>
<td>$\Gamma \neq \langle \Gamma' \rangle^−$</td>
</tr>
<tr>
<td>A structure $\Gamma$ must have a polarity $−$</td>
<td>$\Gamma \vdash \sqsubseteq^\dagger A$</td>
</tr>
</tbody>
</table>

Table 4.1: Encoding of monotonicity and polarity information

To express this encoding it is enough to use a fragment of the logical language of $\text{MCTL}$. In the next section we will show that the use of types outside this fragment could lead to incorrect polarity assignments.

**Definition 4.19.** [Safe Types] To account for NPI distribution and monotone inferences it is enough to mark only downward monotone functions and leave the upward monotone ones unmarked. Therefore, we need to work only with lexical type assignments from the set $\text{FORM}_1$ defined below,

$$\text{FORM}_1 = \text{ATOM} \mid \text{FORM}_1 / \otimes \text{FORM}_1 \mid \otimes \text{FORM}_1 \setminus \text{FORM}_1 \mid \text{FORM}_1 \setminus \text{FORM}_1 \mid \text{FORM}_1 / \text{FORM}_1 \mid \sqsubseteq^\dagger \text{FORM}_1 \mid \sqsubseteq^\dagger \text{FORM}_1 \setminus \text{FORM}_1 \mid \sqsubseteq^\dagger \text{FORM}_1 \setminus \text{FORM}_1 \setminus \text{FORM}_1.$$ 

where $\sqsubseteq^\dagger \text{FORM}_1 / \sqsubseteq^\dagger \text{FORM}_1$ (and $\sqsubseteq^\dagger \text{FORM}_1 \setminus \sqsubseteq^\dagger \text{FORM}_1$) will be used for lexical assignments of negative polarity items. The prefix $\sqsubseteq^\dagger \otimes$ on the argument formula will allow multiple NPI occurrences, whereas on the value formula it will allow the marking of the context where the NPI occurs, the attraction of a licensor and the rejection of an upward monotone function. In Section 4.4.4 we will discuss how the use of this fragment of the logical language effects the possible derivations the system can make.
Based on the selection of this set of formulas we can identify a set of types which properly represent the polarity of the corresponding structure; we refer to them as safe types.

\[
\text{SAFE} = \text{FORM}_1 \mid \Diamond \text{FORM}_1.
\]

Note that the set of safe types is the same as the set of all subformulas of the formulas in FORM$_1$.

**Lemma 4.20.** Given a derivation $D$ of $\Gamma \vdash A$, if $A \in \text{FORM}_1$ and all the formulas in $\Gamma$ are in SAFE, then all nodes in $D$ are labeled by safe types.

**Proof.** The lemma is a consequence of the subformula property of MCTL+Pol [Sza69, Moo97] and the definition of safe types. QED

In the remainder of the chapter we will consider only derivations with safe types, unless explicitly stated otherwise.

Sánchez’ polarity marking algorithm is carried out by the structural rules in Figure 4.2. The [Pol$-$] rule simply distributes markers into the substructures, and [Pol$--$] cancels them out.

![Figure 4.2: Structural Rules of MCTL+Pol.](image)

The structural rule [Pol$-$] is not in the class of those structural rules which guarantee the logical grammar to be in PSPACE, since it increases the length of the structure (Definition 1.15). However, in a normalized derivation [Pol$--$] always precedes [Pol$-$]. This is possible because the two rules are confluent. Therefore, when no other structural rules apply to mode $\neg$, the total maximum space for an MCTL+Pol sequent is linear with respect to the end-sequent, and hence MCTL+Pol is in PSPACE [Moo02]$^5$.

**Proposition 4.21.** [Complexity of MCTL+Pol ] MCTL+Pol is decidable at most in deterministically polynomial space.

### 4.4.1 A Natural Logic based on MCTL+Pol

In Definition 4.23 we will formally define the intuitive encoding of polarity marked structures given in Table 4.1. First we need the definition below.

**Definition 4.22.** [Normal Form for Sequents] A sequent is said to be in normal form if no polarity structural rules can be applied to it.

Note that any sequent $\Gamma \vdash A$ can be normalized by applications of polarity structural rules reaching a unique sequent $\Gamma' \vdash A$ in normal form.

$^5$This complexity property of MCTL+Pol was pointed out to me by Richard Moot.
Chapter 4. Reasoning with Monotone Functions

**Definition 4.23.** [Polarity of Structures] The polarity of a structure is defined in terms of \( \langle \cdot \rangle^- \). Let \( \Delta \vdash B \) be a normalized sequent and \( \Gamma \) a substructure of \( \Delta \),

(a) Let \( \Gamma = \emptyset \), where \( A \in \text{FORM}_1 \). \( \Gamma \) is said to have negative polarity in \( \Gamma \) if \( \Gamma \) is surrounded by no \( \langle \cdot \rangle^- \), and positive polarity otherwise.

(b) Let \( \Gamma = (A)^- \), where \( A \in \text{FORM}_1 \). \( \Gamma \) is said to have positive polarity in \( \Gamma \) if \( A \) is of the form \( \emptyset \), and negative polarity otherwise.

(c) Let \( \Gamma = (\Gamma_1 \circ \Gamma_2) \). \( \Gamma \) is said to have negative polarity in \( \Gamma \) if both \( \Gamma_1 \) and \( \Gamma_2 \) have negative polarity; positive polarity if both \( \Gamma_1 \) and \( \Gamma_2 \) have positive polarity, and undetermined polarity otherwise.

To understand the way polarity is carried out by the structural language, one has to keep in mind that \( \langle \cdot \rangle^- \) is the structural connective corresponding to \( \emptyset \), the operators which we used to denote decreasing monotone argument positions, and that polarity is linked to monotonicity (Definition 4.9).

**Definition 4.24.** [Monotonicity Rules] Given a derivation \( D \) of a sequent \( \Delta \vdash N : C \) in normal form. Let \( M_A \) be a subterm of \( N \) corresponding to a substructure \( \Gamma \) in \( \Delta \) such that \( \Gamma \neq (\Gamma')^- \), viz. in \( D \) there is a subderivation \( D' \) of \( \Gamma \vdash M : A \). Let \( M', M'' \) be two terms such that \( [M''] \leq_A [M] \leq_A [M'] \), and let \( \Gamma_1, \Gamma_2 \) be two structures corresponding to \( M', M'' \), respectively. Then the following inference can be derived.

\[
\frac{\Delta[\Gamma^+] + N : C}{\Delta[\Gamma_1^+] + N : C} \quad \frac{\Delta[\Gamma^-] + N : C}{\Delta[\Gamma_2^-] + N'' : C}
\]

where \( N' = N[M'/M] \) and \( N'' = N[M''/M] \), \( \Gamma_i^+ \) (resp. \( \Gamma_i^- \)) stand for a structure with positive (resp. negative) polarity.

Note how the formal definition of the monotonicity rules, intuitively described earlier in our discussion of LP+EPol, sheds light on the use of the lambda terms to compute the substitution of the marked structures. In the discussion of the examples we will skip the lambda terms and concentrate on the structures on which the inference is performed.

### 4.4.2 MCTL+Pol at Work

As in Sánchez’ approach, to determine the polarity at the sentence level, we start by assigning monotonicity markers to the entries in the lexicon. From the semantic information we know the monotonicity property of a functional lexical entry. We distinguish \( \langle \text{Mon} \rangle \) functions by prefixing their argument with \( \emptyset \). For instance, since \( \text{doubt} \) is \( \langle \text{Mon} \rangle \) in its first parameter, it is of type: \( (np\backslash s)/\emptyset s \). Note that it can be represented by the lambda term \( \lambda x_s.\lambda y_{np}.(\text{doubt} \ x) \ y \), where \( \text{doubt} \) is a downward monotone (functional)
constant in its first argument. Hence the lambda operator abstracts over a variable \( x \) which occurs in a negative position, and by Corollary 4.11 it correctly represents a downward monotone function. Similarly, since \textit{doubt} is an \textit{\u1d40}Mon function in its second argument, the \( y \) is in a positive position and the second abstraction yields an upward monotone function. Both facts are encoded in the logical type assignment. Let us see the rules at work step by step, focusing on the flow of information from the logical to the structural formulas.

- Application of a downward monotone function implies the propagation of the marker from the function to the argument:

\[
\frac{\Delta \vdash B}{\langle \Delta \rangle^{-} \vdash \otimes B} \quad \frac{\Gamma \vdash \otimes B \setminus A}{\langle \Delta \rangle^{-} \circ \Gamma \vdash A} \quad [\\text{E}]
\]

This embodies (part of) Definition 4.9-\textit{iv}: \( M \) is negative (positive) in \( PQ \) if \( M \) is positive (negative) in \( Q \), and \( P \) denotes a downward monotone function. For the missing part (the computation of the polarity of \( M \) in \( Q \)) we need to compute the polarity of the structure corresponding to \( M \) in the structure corresponding to \( Q \). This is done by means of the structural rules.

- Functions are built by applying \([/I], [/I]\). Upward and downward monotone functions abstract over positive and negative positions, respectively, as shown below looking at \([/I]\) by means of an example. Let \( C \neq \otimes C' \),

\[
\begin{align*}
\Gamma \circ C & \vdash A & & [/I] \\
\Gamma & \vdash A/C & & [\\text{E}]
\end{align*}
\]

\[
\begin{align*}
\Gamma \circ \otimes B & \vdash A & & [/I] \\
\Gamma & \vdash A/\otimes B & & [\\text{E}]
\end{align*}
\]

- The polarity information is passed from the structure to the logical type (\textit{a}) and \textit{vice versa} (\textit{b}).

\[
\begin{align*}
\Delta^{-} \vdash A & \quad [\text{\textit{a}}] \\
\Delta \vdash \text{\textit{a}}^{-} A & \quad [\text{\textit{b}}]
\end{align*}
\]

\[
\begin{align*}
\Delta^{-} \vdash A & \quad [\text{\textit{a}}] \\
\Delta \vdash \text{\textit{a}}^{-} A & \quad [\text{\textit{b}}]
\end{align*}
\]

- The substitution of a structure with negative polarity by another one with the same polarity is performed by \([\otimes E]\). The soundness proof (Section 4.4.4) guarantees the correctness of this substitution.

\[
\frac{\Delta \vdash \otimes A \quad \Gamma[\langle A \rangle^{-}] \vdash B}{\Gamma[\Delta] \vdash B} \quad [\otimes E]
\]
We now look at some concrete examples. In Example 4.14, we consider the case of coordinations for the system LP+EPol. Unlike there, we cannot leave (functional) variables unspecified for their monotonicity property, since in MCTL+Pol the monotonicity marking is not assigned ‘externally’. Let us see what differences this will make.

**Example 4.25.** [Coordination] As we have seen, coordination of a noun phrase with a quantifier involves the lifting of the np. In MCTL+Pol the type np can be lifted to a higher order one, by assuming an unmarked type or a marked one.

\[
\frac{\text{mary} \vdash np \quad [x \vdash np]^{1}}{\text{mary} \circ x \vdash s} \quad \frac{\text{mary} \vdash np \quad [\varnothing I]}{[\varnothing I]} \quad \frac{\text{mary} \circ x \vdash s}{[\varnothing I]}^1
\]

The lambda term corresponding to the lifted type is \(\lambda P . P \cdot m\), where \(P\) represents an upward monotone function in (a) and a downward monotone function in (b). Hence \(m\) is in a positive position in (a) and so is its corresponding structure, whereas in (b) it is in a negative position and it is marked by \(\langle \cdot \rangle^{-}\). This lifted type can now be coordinated with a quantifer phrases. The conjunction \(\text{and}\) receives a polymorphic type \((X \backslash X) / X\) which respects the monotonicity properties of the coordinated phrases as illustrated by the derivation below.

\[
\frac{\text{and} \vdash ((s/\varnothing np) / (s/\varnothing np))/\text{every\_boy} \vdash s/\varnothing np}{\text{and} \circ \text{every\_boy} \vdash s/\varnothing np} \quad \frac{\text{mary}^{-} \vdash s/\varnothing np}{[\varnothing I]}^1
\]

The argument of this function must be of type \(\varnothing np\cdot s\), hence it can only be a downward monotone function, let it be \(R\). The lambda term corresponding to the whole structure is \(\lambda Z . (\lambda P . P \cdot m) \cdot Z \backslash \text{Every\_boy} \cdot Z\) \(\cdot R\). By \(\beta\)-reduction the term reduces to \(R \cdot m \backslash \text{Every\_boy} \cdot R\), where \(m\) is in a negative polarity position. Hence, the corresponding structure is properly marked. If \(\text{and}\) had been composed with the derivation in (a) above, the verb phrase could have been only \(\uparrow \text{Mon}\) and again the polarity would have been correctly assigned in the structure. In other words, the type (and therefore the monotonicity property) of the assumed variable depends on the monotonicity property of the expression it will be replaced with.

The example above is also interesting for a second reason. From a closer look at the ‘lexical’ type of \textit{every boy} we see that if we represent the lexical semantics of the constant \textit{every boy} as \(\lambda Q . \forall x . \text{Boy}(x) \rightarrow Q(x)\), the polarity of the second occurrence of \(x\) depends on the monotonicity of \(Q\) (or the term replacing it by \(\beta\)-conversion). In Example 4.25, \(Q\) is replaced by a downward monotone function \(R\), hence \(x\) is in a negative polarity position. The type matching forces \(Q\) to have the same monotonic property. This is reflected by the logical type of the quantifier \(s/\varnothing np\cdot s\) —which would have been \(s/np\cdot s\) in case \(R\) was an upward monotone function. In Chapter 7, we will discuss the behavior of NPIs with respect to coordination.
4.4. Internalizing Monotonicity and Polarity Markers in MCTL

Finally, before looking at how MCTL+Pol can work with structure containing NPIs we want to clarify the relevance of using a selected fragment of the logical language. Note that lexical assignments from outside the set FORM may generate unsafe types in the derivation. The example below shows that this can lead to an incorrect computation of polarity positions. This is why we restrict ourself to a selected fragment of the logical language.

**Example 4.26.** [Unsafe Types] In the derivations (a) and (b) below the types in the boxes are not in SAFE. In (a) the structure $\Delta$ is assigned a negative polarity though it is not in a decreasing argument position. In (b) the structure $\Gamma \circ \Delta$ is assigned a positive polarity, though it is the argument of a downward monotone function.

(a) $$\frac{\Delta \vdash \Box \downarrow (A \setminus B)}{\Gamma \vdash A [\Box \downarrow \mathrm{E}]}
\frac{(\Delta)^- \vdash A \setminus B [\mathrm{E}]}{\Gamma \circ (\Delta)^- \vdash B [\mathrm{E}]}$$

(b) $$\frac{\Gamma \vdash A \quad \frac{\Delta \vdash A \setminus B [\mathrm{E}]}{\Gamma \circ \Delta \vdash B [\mathrm{E}]}}{\Sigma \vdash B \setminus C [\mathrm{E}]}$$

4.4.3 Negative Polarity Items in MCTL+Pol

Because monotonicity information is available on-line during parsing, MCTL+Pol can control NPI distribution. The type assignment of an NPI must encode three pieces of information: (i) the structure where the item occurs must have negative polarity; (ii) the structure containing the NPI must be taken as an argument by a structure of type $A \setminus B$ or $B \setminus A$ (as negative polarity is assigned by downward monotone functions); (iii) NPIs must be in the immediate scope of their licensor. Using the encoding in Table 4.1 this means that the the whole structure containing the NPI must be headed by $\Box \downarrow \top$ and the NPI must have wide scope in the corresponding lambda term. Let us look at the adverb *yet* by means of an example.

**Example 4.27.** [Negative Polarity Items] Let $\Box \downarrow \top \downarrow iv \setminus \Box \downarrow \mathrm{diaminusiv}$ be the type assignment of *yet*. It denotes that it must occur in a negative polarity position which it passes to its argument.

1. Nobody left yet.

$$\frac{\left(\left(\elleft\right)^-\vdash \top \downarrow iv [\top I]\right]}{\left(\left(\elleft\right)^-\vdash \top \downarrow iv [\top I]\right)}$$
$$\frac{\elleft \vdash \Box \downarrow \top \downarrow iv [\Box \downarrow I]\ 
\begin{array}{c}
\text{yet} \vdash \Box \downarrow \top \downarrow iv \quad [\Box \downarrow E]
\end{array}}{\nobody \circ (\elleft \circ \text{yet})^- \vdash s [\top E]}$$

If we replace the quantifier *nobody* with an $\uparrow \mathrm{Mon}$ quantifier, e.g. *everybody*, the derivation fails: *everybody* has type $s/iv$ which does not provide the needed negative feature to match the type assigned to *left yet*. 
Example 4.28. [NPIs in Embedded Sentences]
1. John doubts anybody left.

\[
\begin{align*}
\text{anybody} & \vdash \sqcap_1 \otimes s / \sqcap_1 \otimes iv \\
\text{left} & \vdash \sqcap_1 \otimes iv \\
\text{anybody} \circ \text{left} & \vdash \sqcap_1 \otimes s \\
\text{[E] } & \quad \text{[E] } \\
\text{doubts} & \vdash (np \backslash s) / \otimes s \\
\text{(anybody} \circ \text{left})^{-} & \vdash np \backslash s \\
\text{[E] } & \quad \text{[E] } \\
\text{John} & \vdash np \\
\text{John} \circ (\text{doubts} \circ (\text{anybody} \circ \text{left})^{-}) & \vdash s
\end{align*}
\]

The negative auxiliary didn’t receives a polymorphic type X/X. The different instantiation of the variable type, must encode the monotonicity property of the expression. Furthermore, by decorating the X differently with unary operators, we can control the ways didn’t composes with the bridge and non-bridge predicates.

2. *John didn’t doubt anybody left.

Starting from the lambda term \( \lambda Pxy. \neg (P x) y \) representing didn’t, the phrase didn’t doubt has term \( \lambda xy. \neg (\text{doubt} x) y \) and hence P is ↓Mon. The polarity of x is positive since it is in the scope of two downward monotone constants \( \neg \) and \( \text{doubt} \). Therefore, the function \( \lambda xy. \neg (\text{doubt} x) y \) is ↑Mon. This makes the whole sentence semantically ill-formed, which is correctly predicted by \( \text{MCTL}+\text{Pol} \): By simply applying our encoding, the type for didn’t is \( (iv/s) / \otimes (iv / \otimes s) \) from which it follows that the phrase didn’t doubt is of type \( (iv/s) \) which fails to compose with anybody left.

3. John didn’t think anybody left.

Similarly to the example above, the monotonicity properties of think and didn’t motivate the type assignments of the items involved here, and in particular of didn’t. The term \( \lambda xy. \neg (\text{think} x) y \) represents a downward monotone function, since think is an upward monotone function. Therefore, type matching requires the assignment to didn’t of a term \( \lambda Pxy. \neg (P x) y \) where P denotes an ↑Mon function, and consequently the type \( (iv/ \otimes s) / \otimes (iv / s) \). The composed function ‘didn’t think’ of type \( (iv/ \otimes s) \) licenses the occurrence of the NPI anybody.

Example 4.29. [Multiple Negative Polarity Occurrences] Multiple occurrences of negative polarity items can be licensed by the same licensor, as illustrated by anybody and at all and their trigger doubts.

1. John doubts anybody came at all.

\[
\begin{align*}
\text{anybody} \circ \text{came} & \vdash \sqcap_1 \otimes s \\
\text{at all} & \vdash \sqcap_1 \otimes s / \sqcap_1 \otimes s \\
\text{doubts} & \vdash (np \backslash s) / \otimes s \\
\text{(anybody} \circ \text{came}) \circ \text{at all} & \vdash \sqcap_1 \otimes s \\
\text{[E] } & \quad \text{[E] } \\
\text{John} & \vdash np \\
\text{John} \circ (\text{doubts} \circ ((\text{anybody} \circ \text{came}) \circ \text{at all})^{-}) & \vdash s
\end{align*}
\]
We close this section with an example that is still problematic for our system as it stands—we return to it in Chapter 6. The problem relates to generalized quantifier expressions, which in natural language can have varying scopal possibilities. For instance, the sentence *Three good referees read few abstracts* can be assigned two different representations: either 

(Three\_good\_referees \ Few\_abstract \ Read) or (Few abstracts \ Three\_good\_referees \ Read). The different scope of the downward monotone function *Few abstract* modifies the polarity of the subterm in the whole term. In the first case, the constant *Three good referees* has positive polarity; whereas in the second case, it is in negative positions. From this it follows that different inferences will be drawn from the sentence *Three good referees read few abstracts*, depending on the interpretation assigned to it.

(a) 

(Three good referees)\(^+\) read few abstracts \(\vdash_s\) 

Three referees read few abstracts \(\vdash_s\)

(b) 

(Three good referees)\(^−\) read few abstracts \(\vdash_s\) 

Three good dutch referees read few abstracts \(\vdash_s\)

where (a) would be logically correct in the interpretation (Three > Few) and (b) in the wide scope reading (Few > Three). However, while (a) is a correct natural reasoning inference, (b) is not. This example shows that natural reasoning, and in particular monotone inference, also uses other information than the semantic information discussed so far. Though monotonicity plays a crucial role in the derivation of inference, properties of different nature may also interact with it and affect natural reasoning inference. In Chapter 6, we will explore the properties affecting the interpretation of linguistic structures containing quantifier phrases (such as the one in the premises of the inferences above) and give an deductive analysis within the CTL framework.

### 4.4.4 Soundness

We have to guarantee that our natural logic does not derive inferences which are not valid model theoretically, i.e. we need to prove that the natural logic is sound.

We use \(\phi \vdash_{\text{MON}} \psi\) to denote that a marked parsed structure \(\psi\) is inferred from the marked structure \(\phi\) by means of the monotonicity rules given in Definition 4.24. Formally, proving that the natural logic fragment is sound means to prove that for all models \(\mathcal{M}\) and assignments \(f\),

\[
\text{if } \phi \vdash_{\text{MON}} \psi \text{ then } [\phi]_f^\mathcal{M} \leq [\psi]_f^\mathcal{M}
\]

where \([\phi]_f^\mathcal{M}\) and \([\psi]_f^\mathcal{M}\) stand for the denotations of the lambda terms corresponding to the structures \(\phi\) and \(\psi\) in \(\mathcal{M}\) under \(f\), respectively.

Proving the above claim reduces to proving that monotonicity in lambda terms is tied up with the syntactic notion of polarity, and consequently to the markers assigned to the parsed structures. The link between monotonicity and polarity is given by Proposition 4.10 and its Corollary 4.11 as we have already commented. Our task is now to
prove that if a substructure in the conclusion of a (sub)derivation has positive (resp. negative) polarity, then the corresponding lambda term is in a positive (resp. negative) position in the term corresponding to the whole conclusion (see Definition 4.8).

**Proposition 4.30.** [Soundness] Let \( D \) be a derivation of \( \Gamma \vdash M : A \) with \( A \in \text{SAFE} \) and \( \Gamma \) built on \( \text{SAFE} \). Let \( (B/C)/D \) denote \( (((B/C)/D_1) \ldots D_n) \) with \( n \geq 0 \). Then\(^7\),

(a) If \( A \) is of the form \( (B/C)/D \) or \( \ominus((B/C)/D) \) (modulo occurrences of \( \ominus \) \( \ominus \)), then \( M \) is \( \downarrow\text{Mon} \) in the argument corresponding to \( C \) if \( C \) is of the form \( \ominus C' \), and \( \uparrow\text{Mon} \) otherwise.

(b) If \( \Gamma \) contains a logical formula \( (B/C)/D \) or \( \ominus((B/C)/D) \) (modulo occurrences of \( \ominus \) \( \ominus \)), then the lambda term in \( M \) corresponding to this substructure is \( \downarrow\text{Mon} \) in the argument corresponding to \( C \) if \( C \) is of the form \( \ominus C' \), and \( \uparrow\text{Mon} \) otherwise.

(c) If \( A \in \text{FORM}_1 \), then for any substructure \( \Gamma' \) in \( \Gamma \), if \( \Gamma' \) has positive (resp. negative) polarity in \( \Gamma \) then the corresponding lambda term is in a positive (resp. negative) position in \( M \).

(d) If \( A \) is of the form \( \ominus A' \), then for any substructure \( \Gamma' \) in \( \Gamma \), if \( \Gamma' \) has negative (resp. positive) polarity in \( \Gamma \) then the corresponding lambda term is in a positive (resp. negative) position in \( M \).

**Proof.** The proof goes by induction on the length of the derivation.

1. Base case. Assume \( \Gamma \vdash M : A \) is a leave. Then (a), (b), (c) and (d) hold trivially.

2. I.H. Let \([R]\) be the last rule applied. Assume (a), (b), (c) and (d) hold for the premises of \([R]\), we have to show that they hold also in its conclusion.

   (i) (i\(_1\)) \([R] = \text{[Pol\_]; [Pol\_]}.\)

   Neither the polarity of the structures, nor the lambda term, nor the formula on the right side of the \( \vdash \) changes. Hence, the proposition holds by I.H.

   (ii) (ii\(_1\)) \([R] = \text{[\Theta I]; [\Theta I]; [\Theta I]; [\Theta I].}\)

   (ii\(_1\))

   \[
   \begin{array}{c}
   D_1 \\
   \vdots \\
   \Gamma' \vdash M : A' \\
   (\Gamma' \vdash M') \ominus A' \Theta I
   \end{array}
   \]

   (a) and (b) are preserved\(^8\). Since \( A \in \text{SAFE} \), we know that \( A' \in \text{FORM}_1 \). By I.H. (c) holds in the premise, hence (d) holds in the conclusion of \([\Theta I]\).

   The same applies in case \([R] = [\Theta I], \) or \([R] = [\Theta I]$.$}

\(^7\)We simplify the proof by considering only \( \vdash \). The directionality of the functional implication does not affect the proof.

\(^8\)The terms \( M \) and \( \ominus M \) have the same monotonicity property. The same holds for terms decorated by the other unary operators introduced by the logical rules of \( \Theta \) and \( \ominus \), \( \cup \), \( \vee \) and \( \wedge \).
(iii) \([R] = \lceil /I \rceil\).

\[
\begin{array}{c}
D_1 \\
\vdots \\
\Gamma \circ C \vdash M : B \\
\Gamma \vdash \lambda x. M : B/C \\
\end{array} \quad \lceil /I \rceil
\]

(a) Let \(B', C', \overline{D}\) be such that \(A = ((B'/C')/\overline{D})\). If \(n \geq 0\), then (a) follows directly from I.H. Otherwise \(B' = B\) and \(C' = C\). Since \(B/C \in \text{SAFE}\), \(B \in \text{FORM}_1\). Hence (c) applies to the premise of \(\lceil /I \rceil\). Therefore, the lambda term corresponding to \(C\) is in an positive (resp. negative) position in \(M\), if \(C \neq \ominus C''\) (resp. \(C = \ominus C''\)). By Corollary 4.11, \(\lambda x. M\) denotes an \(\lceil \text{Mon} \rceil\) (resp. \(\lceil \text{\lnot Mon} \rceil\)) function in \(x\) (the variable corresponding to \(C\)).

The points (b) and (c) follow directly by I.H., and (d) holds trivially.

(iv) \([R] = \lceil /E \rceil\).

\[
\begin{array}{c}
D_1 \\
\vdots \\
\Delta \vdash t : A/B \\
\Gamma \vdash u : B \\
\Delta \circ \Gamma \vdash t(u) : A \\
\end{array} \quad \lceil /E \rceil
\]

The points (a) and (b) apply by I.H. and (d) holds trivially. We have to consider (c).

Let \(\Gamma'\) be in \(\Delta \circ \Gamma\), we have to consider three cases: 1. \(\Gamma'\) is in \(\Delta\), 2. \(\Gamma'\) is in \(\Gamma\), and 3. \(\Gamma' = \Delta \circ \Gamma\).

1. If \(\Gamma' \subseteq \Delta\), then (c) applied to \(\Gamma'\) follows directly from I.H. applied to the major premise.

2. If \(\Gamma' \subseteq \Gamma\), then
   (2') Suppose \(\Gamma'\) has positive polarity in \(\Gamma\), then it has positive polarity also in \(\Delta \circ \Gamma\). If \(B \in \text{FORM}_1\), then by (a) \(t\) denotes an \(\lceil \text{\lnot Mon} \rceil\) function, and by (c) applied to the minor premise of \(\lceil /E \rceil\), the lambda term corresponding to \(\Gamma'\) occurs in an positive position in \(u\). By Definition 4.9, it is in an positive position in \(t(u)\). If \(B = \ominus B'\), then by (a) \(t\) denotes a \(\lceil \text{\lnot Mon} \rceil\) function, and by (d) applied to the minor premise of \(\lceil /E \rceil\), the lambda term corresponding to \(\Gamma'\) occurs in a negative position in \(u\). By Definition 4.9, it is again in an positive position in \(t(u)\).
   (2'') Similarly for \(\Gamma'\) with negative polarity in \(\Gamma\).

3. If \(\Gamma' = \Delta \circ \Gamma\), then (c) holds trivially.

(v) \([R] = \lceil \ominus E \rceil\).

\[
\begin{array}{c}
D_1 \\
\vdots \\
\Delta \vdash u : \ominus A \\
\Gamma \vdash (v : A) \vdash t : B \\
\Gamma[\Delta] \vdash t[v/u] : B \\
\end{array} \quad \lceil \ominus E \rceil
\]
First of all, observe that by I.H. on (a) and (b), the monotonicity property of \( u \) and \( v \) are both completely determined by the form of \( A \) in exactly the same manner. Thus, substituting \( u \) for \( v \) in \( t \) does not affect any monotonicity related property. Let us refer to this as \((*)\).

(a) The claim in (a) follows directly from \((*)\) and the I.H. on the minor premise of \([\Diamond E]\).

(b) For \( \Gamma' \) in \( \Gamma \) (b) applied to \( \Gamma' \) follows from \((*)\) and the I.H. on the minor premise. For \( \Gamma' \) in \( \Delta \) (b) follows directly by I.H. on the major premise of \([\Diamond E]\).

(c) \( B \in \text{FORM}_1. \) If \( \Gamma' \) in \( \Gamma \) or \( \Gamma' = \Gamma''[\Delta] \), then (c) applied to \( \Gamma' \) follows from \((*)\) and the I.H. on the minor premise of \([\Diamond E]\). For \( \Gamma' \) in \( \Delta \),

1. If \( \Delta \) is surrounded by an odd number of \( \langle \cdot \rangle^- \) in \( \Gamma[\Delta] \), then \( A \) has positive polarity in the minor premise (it is surrounded by an even number of \( \langle \cdot \rangle^- \)). Then by I.H. on (c) \( v \) is in an positive position in \( t \). Furthermore, if \( \Gamma' \) in \( \Delta \) has positive polarity in \( \Delta \) then it has negative polarity in \( \Gamma[\Delta] \). By I.H. on (d) applied to \( \Gamma' \), the term \( z \) corresponding to \( \Gamma' \) in \( u \) is in a negative position, and by Definition 4.9, since \( v \) is in positive position in \( t \), \( z \) is in a negative position in \( t[u/v] \). Similarly, for \( \Gamma' \) with negative polarity in \( \Delta \).

2. If \( \Delta \) is surrounded by an even number of \( \langle \cdot \rangle^- \) in \( \Gamma[\Delta] \), then \( A \) has negative polarity. The proof is similar to the previous case, apart from the fact that \( v \) is in a negative position and \( A \) is surrounded by an odd number of \( \langle \cdot \rangle^- \).

(d) \( B = \Diamond B \), the proof is similar to the one of case (c).

\[ \text{QED} \]

### 4.4.5 Summary

We have already commented on some of the differences between the three systems \( \text{LP+EPol}, \text{CG+Pol} \) and \( \text{MCTL+Pol} \), the most important being that the first system obtains polarity markers \textit{extra-logically}, while in \( \text{CG+Pol} \) and \( \text{MCTL+Pol} \) the marking is obtained on-line as part of the logical derivations. Now let us analyze carefully the effects that these differences have on the final aim of the project: the design of a natural logic to account for negative polarity items.

To start with, all three formalisms should be able to determine the grammaticality of linguistic structures. This is a basic requirement, even if we are not interested in analyzing monotonicity phenomena. Here already, \( \text{LP+EPol} \) and \( \text{CG+Pol} \) are outperformed by \( \text{MCTL+Pol} \), since \( \text{LP} \) does not take syntactic structure into account, while \( \text{CG+Pol} \) lacks hypothetical reasoning and needs multiple lexical entries to compensate for this deficiency (Chapter 1).

More interesting to the subject discussed in this chapter is the difference in dealing with polarity intra-logically and extra-logically. \( \text{LP+EPol} \) implements polarity marking as a rewriting algorithm which takes a proof tree as input and decorates it with markers.
4.5. Key Concepts

It is crucial to note that this approach leaves out the possibility of polarity information actually taking active part in the derivation. In CG+Pol and MCTL+Pol, in contrast, the monotonicity information stored in the lexicon plays a fundamental role during the construction of the derivation. The presence of monotonicity markers enables or blocks the possibility of applying specific derivation rules. In other words, beside working as markers for making valid inferences at the natural reasoning level, they contribute to determine grammatical structures.

Moreover, the analysis of NPIs in embedded sentences shows that the way monotone functions compose in natural language can be effected by other constraints, and some control on functional application is necessary in order to properly model this linguistic phenomenon. Neither LP nor CG has the right expressivity to tackle this problem. On the other hand, MCTL provides us with the required logical tool kit to obtain fine-grained type assignments that lexically anchor the way structures are built.

Furthermore, there are also other things to be gained by analyzing polarity phenomena in MCTL. Negative polarity phenomena are observed in many languages. Being able to account for them in terms of the base logic opens the door to a comparative study of both NPIs and natural reasoning. Crosslinguistic variations could be accounted for in terms of different type assignments and structural rule packages. An interesting area for future research is how structural rules interact with the computation of the polarity positions and how they effect the distribution of negative polarity items. LP+EPol is too flexible, and CG+Pol is too strict to tackle this question.

Finally, a last remark concern the correctness of the marking mechanisms used by the three systems. While LP+EPol and MCTL+Pol are proven to be sound, no analogous result is known for CG+Pol. Note that the soundness of the marking algorithm does not fully guarantee the correct modelling of natural reasoning inference, since natural language may diverge from formal language in the combinatorial scope possibilities of its scoping elements. An important constraint the parser must satisfy is therefore that it produces as output only the available readings of the parsed linguistic structure. For the reasons discussed so far MCTL+Pol seem to be most promising in this respect.

4.5 Key Concepts

The analysis discussed in this chapter shows that:

1. Monotonicity and polarity information can be encoded into CTL syntactic type assignments by means of unary operators.

2. CTL can be employed to achieve a proof theoretical account of semantic issues.

3. A system in which semantic information relevant for grammaticality is internalized into the logical language presents several advantages with respect to a system based purely on functional types where semantic information is post hoc by and extra-logical marking mechanism. In particular, it can account for the distribution of items sensitive to such information.
4. MCTL seems to provide a natural division of labor between the tasks of reasoning and parsing. The latter is handled in the logical part of a sequent, the former in the structural part.