Utrecht University

Incrementally interpreting wh-questions using typelogical grammars

by

Pim Hopmans, 4190114

Abstract

Incremental interpretation is a field that is concerned with building semantic terms of natural language sentences word-by-word from left-to-right. This field could be very relevant for linguistics and artificial intelligence as psycholinguistic evidence has pointed out that humans most likely interpret language incrementally. This thesis is focused upon the incremental interpretation of questions that start with wh-phrases, such as "who", "what" and "which", i.e. wh-questions. As a framework for interpretation, a non-associative typelogical grammar is used that makes use of controlled associativity and reordering. This is a strongly lexicalised formalism, where every word is assigned a type and semantic term depending on the context that they occur in. On top of the typelogical grammar, the system **M** due to Moortgat [1988] is used. This system combines any two types of the grammar, including types for wh-phrases, to a single type. The semantic terms at each point also follow from this system. For our purposes, **M** is translated to a system **M**♦ that accommodates controlled associativity and reordering. **M**♦ is consequently used within an incremental interpretation algorithm and restricted it to always take the first two types of a wh-question as input. This results in incremental interpretation of wh-questions.

INFOMAI2
Master's Thesis (44 ECTS)
July 2019
Supervisor: Dr. Rick Nouwen
Examiner: Prof. dr. Michael Moortgat

Contents

Chapter 1

Introduction

Incremental interpretation has not been a very big subject of research until now. It is, however, very relevant to research fields such as linguistics and artificial intelligence. Psycholinguistic evidence has been found that state that humans interpret natural language incrementally (Marslen-Wilson [1973], Just and Carpenter [1980], Altmann and Steedman [1988], Kamide and Haywood [2003], Kuperberg et al. [2003]). That is, humans maintain a representation of the structure and semantic value of the sentence, even when the sentence has not been completely uttered yet. For each word of a sentence a human has been given as input, it is connected to the maintained structure and the semantic value of the entire sentence is computed.

One of the benefits of incremental interpretation is that it does not require a sentence to be uttered completely before interpretation starts. Milward and Cooper [1994] explain that a sentence such as the following has 4862 syntactical parses:

I put the bouquet of flowers that you gave me for Mothers' Day in the vase that you gave me for my birthday on the chest of drawers that you gave me for Armistice Day.

Incremental interpretation is able to lower the number of possible parses as it can strike out any parse that is not plausible as soon as a certain structure is found. Because of this benefit, incremental interpretation could be linear in complexity as opposed to non-incremental interpretation. Since non-incremental interpretation requires the sentence to be uttered completely before interpretation, the search space will be larger when parsing a sentence and as a consequence, the complexity will also be higher. This holds even for relatively small and simple sentences, such as *"John thinks that Mary saw a bird."*.

Also, incremental interpretation gives humans the ability to predict what their conversation partner is trying to say, as in the following dialogue:

Person A: *I think I have a pretty extensive, uh...*
Person B: *Vocabulary?*
Person A: *Yes!*

Concerning the field of artificial intelligence, Milward and Cooper explain that incremental in-

terpretation could be useful when feedback should be redirected immediately to the user of an interface that uses speech recognition. In this case, they referred to the Foundations of Intelligent Graphics Project which applied incremental interpretation to the utterances of users of a kitchen design system. During the utterance of the user, some part of the interface of this project got highlighted to provide confirmation or to steer the user into giving more information. Naturally, this could be expanded on a broader level, in the sense that any intelligent agent might provide any kind of feedback during the utterances of users to help them reach their goals.

Lastly, Milward and Cooper provide an example of dialogue processing where incremental interpretation is useful to deal with interruptions of utterances, such as hesitations, replacements and insertions. For example, in the following sentence, incremental interpretation is able to connect the antecedent *"the three main sources of data"* with the pronoun *"they"*:

The three main sources of data come, uh ..., they can be found in the references

Until now, the examples have only been declarative sentences. The incremental interpretation of questions, however, might be more interesting to artificial intelligence as computing the response can already start whenever a question is being uttered. This could help reduce the response time of any intelligent system. In this thesis, the scope of the research is narrowed down to one particular type of question, namely wh-questions. Wh-questions are questions that contain wh-phrases, such as *'who'*, *'what'*, *'where'*, *'which'*, etc. Wh-questions are particularly interesting subjects for interpretation as they are open questions, i.e. questions that do not specify the possible answers to it. However, the wh-phrase that is fronted does set some constraint on what kind of responses count as answers. The wh-phrase *'who'*, for example, specifies that the answer to the question must be a person or entity. The incremental interpretation of such questions therefore provides a lot of information early on during interpretation. Wh-phrases do not give any clue as to what answer is the true answer, so in this thesis, we will not discuss what a correct answer to a wh-question should be. We are merely interested in how incremental interpretation can be modeled. Furthermore, as there exists a variety of wh-phrases, we will not discuss all of them. In this thesis, we will only treat argument wh-phrases, such as *'who(m)'*, *'what'* and *'which'*.

Analogously to the interpretation of declarative sentences, the answer to a wh-question may very well be thought of by a person before his/her conversation partner has finished the question. For example, the wh-phrases *'who'* and *'what'* indicate that the answer to the question will be a person or an event/object, respectively, next to the fact that they indicate the utterance being a question in the first place. A wh-determiner such as *'which'* doesn't give such an indication towards the answer, but awaits a predicate such as *'woman'* to indicate what the answer might be. The information hidden in wh-phrases can therefore be very useful for different applications and, in the ideal case, should be reflected in the semantic term corresponding to them.

In our perception, the most obvious application of incremental interpretation is the fact that searching the answer space of a certain wh-question can start before the question is completely uttered. The information that wh-phrases and predicates provide constraints on the answer space corresponding to the question, and as a consequence make the answer space smaller. Complexity wise, an algorithm that searches for a correct answer to the question takes less time when the answer space is smaller. Continuing on the example of the wh-phrase *'who'* that specifies that

the answer to the question should be a person, we might think of the following scenario: A speaker is at a party. At this party, cake was served and at the moment all cake has been eaten. The speaker is wondering who ate the last piece of cake and asks a friend who it was. We make a snapshot in the middle of the utterance of the speaker and end up with the following situation:

Full sentence: *Who in this room ate the last piece of cake?*
Snapshot: *Who in this room ate...*

At the moment of this snapshot, the word *"who"* has indicated that a question will be formed and an answer is required. On top of that, it has indicated that the answer space will only contain persons. In its turn, the succeeding phrase *"in this room"* constrains the answer space to only contain persons that are currently in the same room as the speaker. The result of that constraint is a new, smaller answer space. Subsequently, the verb *"ate"* places a constraint on the new answer space to only contain people that actually ate something. Thus, at the time of the snapshot, the friend of the speaker will already know that the answer to the question must be a person in the same room that has eaten something. He/She can already start thinking about who would be the person the speaker is interested in, even though he/she hasn't yet heard what that person should've been eating.

This example is still in the context of human interpreting, but this could be analogous to, for example, a virtual assistant. Say, a person has a virtual assistant and wants to know who has discovered the continent of America, the following situation could take place:

Full sentence: *Who has discovered the continent of America?*
Snapshot: *Who has discovered...*

At the moment of this snapshot, the virtual assistant could already be searching for anyone who has ever discovered something. At the moment that *"the continent"* is added to the question, the search space is constrained even further and the answer should be given relatively quickly after the question has been uttered when compared to if computation would only start after the question has been completely uttered. So, connecting the phrases in real time to each other, could upgrade the performance of such virtual assistants.

Of course, we have to use a linguistic framework to model incremental interpretation. For this thesis we have chosen to model incremental interpretation with a typelogical grammar [Moortgat, 1997]. The strength of this grammar is that it is a strongly lexicalised categorial system. The interpretation of natural language is fully driven by the types assigned to the smallest lexical elements, namely words. Moreover, in the ideal world every word has exactly one type, which means that for interpretation of different sentences in different contexts, the same set of words and types can be used. Typelogical grammars are chosen for this thesis, as they have a clear intuition about words and the syntactical context they appear in. Adding to that, the semantic value of a sentence follows easily from the syntactical derivation. It is a very useful framework to model natural language and therefore very useful for AI as well. Interpretation of wh-questions with typelogical grammars has been explored by Vermaat [2006]. She formulates wh-type schemata for different kinds of wh-phrases and how they should be interpreted. The system of incremental interpretation this thesis presents is mostly inspired by her work.

This thesis is therefore meant to answer the question: is it possible to model incremental inter-

pretation of wh-questions using a typelogical grammar? And if it is possible, what is the best way to model incremental interpretation? We will find that it is indeed possible, thanks to a system **M** originally due to Moortgat [1988]. Moreover, it makes incremental interpretation of wh-questions quite straightforward. The thesis is divided into four parts. Chapter 2 will lay the grounds of the typelogical grammar that is used in the remainder of this thesis. Chapter 3 will elaborate on what kind of questions exist and why wh-questions are the topic of this thesis. Furthermore, it will present the wh-type schemata originally due to Vermaat [2006]. Chapter 4 will dive deeper into the concept of incrementality and show how it has (or has not) been modeled in other frameworks. Chapter 5 will present how the problem of achieving incrementality with the typelogical grammar has been pieced together by means of extending the typelogical grammar with a variant of **M**. And we will end with a discussion of the system that has been presented and where it might be extended even further in Chapter 6.

Chapter 2

Typelogical Grammar

2.1 Introduction

A typelogical grammar [Moortgat, 1997] is a linguistic framework that provides the means to model a variety of linguistic phenomena. It is a strongly lexicalised categorial system, meaning that typelogical grammar is fully driven by the types given to lexical elements: words. The types are built up out of a few basic types and a set of operators, either binary or unary, operating on the basic types. The type of a word indicates how it composes with the words surrounding it and therefore classifies the word as belonging to a certain category of words. For instance, all intransitive verbs behave in the same manner grammatically, in the sense that they compose with a noun phrase on their left-hand side to form a sentence. Therefore, all intransitive verbs have the same type. To derive whether a phrase is grammatical, a typelogical grammar makes use of derivation rules. Each operator comes with 2 derivation rules, namely its introduction and elimination rule. The derivation rules are not only able to derive the grammaticality of a sentence, but also the semantic value of the sentence. The derivation rules are extended with semantic values which are presented in lambda-calculus. Next to the derivation rules, there exist structural postulates that do not affect the semantics of a derivation, but only the syntactical aspect.

This section will present a typelogical grammar in a gradual manner. First, the type system will be elaborated on. Then, the derivation rules will be presented. And lastly, a few example derivations of natural language sentences will be given to illustrate the machinery of typelogical grammar. All derivation rules can be found in Appendix A for easy reference.

2.1.1 Types and operators

As stated above, a typelogical grammar is driven by the types (or categories) assigned to words. The set of basic (or atomic) types usually consists of the types n for nouns, np for noun phrases and s for sentences. However, the set of basic types is not limited to these types. Later in this thesis we will see wh for wh-questions and q for yes/no-questions. The set of operators consists of $\Box$ and $\Diamond$ as unary operators and the forward slash, /, the backward slash, \, and the product, $\bullet$, as binary operators. The following inductive definition characterizes the full set of types:

Definition 1 (Types/Categories)

$$\mathcal{F} ::= \mathcal{A} \mid \mathcal{F}/\mathcal{F} \mid \mathcal{F} \bullet \mathcal{F} \mid \mathcal{F}\backslash\mathcal{F} \mid \Box\mathcal{F} \mid \Diamond\mathcal{F}$$

Note that a type may be a basic type, but also a complex type, such as $(np\backslash s)/np$.

Considering the binary operators, a type containing a slash, B/A (resp. $A\backslash B$), indicates an expression that is incomplete with respect to an expression of type A to it's right (resp. left). When providing an expression of type A on the right side (resp. left side), the resulting expression will be of type B. The product operator, $\bullet$, indicates a combination of two types. An expression of type $np\backslash s$ results in an expression of type s when an expression of type np is provided to the left, as shown in the following sequent in Gentzen-sequent style presentation:

$$np \bullet np\backslash s \vdash s$$

The sequent-style presentation is commonly used within the typelogical framework and this thesis will also make use of this kind of presentation. The sequent above indicates that a type $np \bullet np\backslash s$ evaluates to type s. In natural language terms, it shows that a noun phrase of type np and an intransitive verb of type $np\backslash s$ in that order is of type s. Examples of these sentences are *"Mary slept"*, *"John swam"* or *"Bill ran"*.

The unary operators, $\Box$ and $\Diamond$, are placed upon a type (either basic or complex) to indicate a special property or feature of that type. The placing of a unary operator upon a type is also called *decorating*. The type $A/\Box B$ now requires an expression of type $\Box B$ on its right and makes explicit that the expression that will be supplied possesses an additional feature. In this thesis, the unary operators will be used to express structural control over certain phrases. In this sense, the $\Box$ and $\Diamond$ will work together as duals of each other to denote a *key-lock* pair, where the $\Diamond$ will lock a constituent out of structural displacement and the $\Box$ will unlock that constituent, as we will see later when their derivation rules are presented. It is important to note that unary operators have a higher precedence than binary operators.

Whenever a typelogical grammar is used, a lexicon will usually be provided to translate between natural language phrases and the syntactic types belonging to those phrases. The '::' operator is used to indicate type-assignment. A lexicon may, for instance, look as follows:

John	:: np	**the**	:: np/n
slept	:: $np\backslash s$	**man**, **apple**	:: n
ate	:: $(np\backslash s)/np$	**somebody**	:: $s/(np\backslash s)$
he	:: $\Box_{sg}np$	**sings**	:: $\Box_{sg}np\backslash s$
they	:: $\Box_{pl}np$	**sing**	:: $\Box_{pl}np\backslash s$

In this lexicon a natural language phrase is connected to a syntactic type as we discussed above. For example, *'John'* is of type np and *'slept'* of type $np\backslash s$. Note that a box feature is added to the words *'he'* and *'they'* to indicate whether it is a singular or a plural noun phrase. Similarly, a box feature is added to the words *'sings'* and *'sing'* to indicate whether these verbs require a singular or plural noun phrase to their left. The semantic value of words are normally also included in a lexicon, but in this case they are omitted for illustrational purposes. Later on, the semantic values will be added.

2.1.2 Derivation rules

To check whether a sentence is grammatical given a lexicon, typelogical grammar makes use of a derivational system. In this derivational system, so-called derivation rules are applied to (complex) types in order to create a 'new' type. A derivation is correct when the complex type of a phrase can be derived from a number of axiom sequents of the form $A \vdash A$ using the available derivation rules. In this section, the derivation rules will be presented along with a general intuition of their function.

Before we go on to presenting the derivation rules, we need a better notion of what a sequent is. As stated above, this thesis uses the sequent-style presentation originally due to Gentzen. This style of presentation claims that *derivability* is a relation between structures and types. Types are already defined above, so the definition of structures is next:

Definition 2 (Structures)

$$Struc \ ::= \ \mathcal{F} \mid Struc \circ Struc \mid \blacklozenge Struc$$

A structure will then be inductively built up out of types and will either combine with another structure via the structural product, $\circ$, or get a structural feature, $\blacklozenge$. The structural product is a binary operator so whenever more than two structures are combined, parentheses are important, e.g. $A \circ (B \circ C)$ is a different structure than $(A \circ B) \circ C$. Vermaat [2006] defined a sequent as follows:

Definition 3 (Sequents) *A sequent is a statement* $\Gamma \vdash A$ *with* $\Gamma \in Struc$ *and* $A \in \mathcal{F}$ *expressing the judgment that the structure* Γ *can be shown to be of type* A*.*

$$Struc \vdash \mathcal{F}$$

Now that we have a proper definition of sequents, it's time to move on to the derivation rules. The derivation rules always follow the same schematic form. They have one or more sequents as premise(s) and will, by introducing or eliminating a specific operator, result in a single sequent. The form of the rules is as follows:

$$\frac{Seq_1 \dots Seq_n}{Seq} \ [Label]$$

In this form we see the premises, $Seq_1 \dots Seq_n$, on top and the conclusion, Seq, on the bottom. $Label$ is an identifier for the rule name. The rule names specify on which operator the rule is applied and whether it is introduced (I) or eliminated (E) in Seq. Each operator in the grammar will now get 2 corresponding rules, one for the introduction of the operator, and one for the elimination of that operator. The definitions we use are originally due to Lambek [1958][1]and are shown in Figure 1. We'll present all definitions of the derivation rules at once, after which we will discuss them individually.

[1]The system Lambek proposed was written down differently, but is essentially the same as the system used for this thesis.

Definition 4 (Axiom rule)

$$\frac{}{A \vdash A}\ Ax$$

Definition 5 (Elimination rules for / and \) *If the structure* Γ *has type* A/B *(or* $B\backslash A$*) and the structure* Δ *has type* B*, then the composition of these two structures,* $\Gamma \circ \Delta$ *(or* $\Delta \circ \Gamma$*), is of type* A*.*

$$\frac{\Gamma \vdash A/B \quad \Delta \vdash B}{\Gamma \circ \Delta \vdash A}\ [/E] \qquad \frac{\Delta \vdash B \quad \Gamma \vdash B\backslash A}{\Delta \circ \Gamma \vdash A}\ [\backslash E]$$

Definition 6 (Elimination rule for •) *Let* Δ *be a structure which has been shown to be of type* $A \bullet B$*, and let* Γ *be a structure with a substructure* $A \circ B$*, where* Γ *has been shown to be of type* C*. Substituting* Δ *for* $A \circ B$ *in the designated position produces a structure which is also of type* C*.*

$$\frac{\Delta \vdash A \bullet B \quad \Gamma[(A \circ B)] \vdash C}{\Gamma[\Delta] \vdash C}\ [\bullet E]$$

Definition 7 (Introduction rules for / and \) *Suppose we have shown for a structure* $B \circ \Delta$ *(or* $\Delta \circ B$*) that it is of type* A*. We can conclude that* Δ *by itself is of type* $B\backslash A$ *(or* A/B*).*

$$\frac{\Delta \circ B \vdash A}{\Delta \vdash A/B}\ [/I] \qquad \frac{B \circ \Delta \vdash A}{\Delta \vdash B\backslash A}\ [\backslash I]$$

Definition 8 (Introduction rule for •) *Merging a structure* Γ *of type* A *with a structure* Δ *of type* B *results in a structure* $\Gamma \circ \Delta$ *of type* $A \bullet B$*.*

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma \circ \Delta \vdash A \bullet B}\ [\bullet I]$$

Definition 9 (Derivation rules for □) $[\Box E]$*: If a structure* Γ *is shown to be of type* $\Box A$*, we may conclude that the structure* $\blacklozenge(\Gamma)$ *is of type* A*.* $[\Box I]$*: The introduction of the* $\Box$ *is obtained by switching premise and conclusion of the elimination rule for the* $\Box$*.*

$$\frac{\Gamma \vdash \Box A}{\blacklozenge(\Gamma) \vdash A}\ [\Box E] \qquad \frac{\blacklozenge(\Gamma) \vdash A}{\Gamma \vdash \Box A}\ [\Box I]$$

Definition 10 (Derivation rules for ◇) $[\Diamond E]$*: Let* Δ *be a structure of type* $\Diamond A$*. If* Γ *is a structure of type* B *containing a substructure* $\blacklozenge A$*, one can substitute* Δ *for* $\blacklozenge A$ *at the designated position in* Γ *to obtain a structure of type* B*.*

$$\frac{\Delta \vdash \Diamond A \quad \Gamma[\blacklozenge(A)] \vdash B}{\Gamma[\Delta] \vdash B}\ [\Diamond E]$$

$[\Diamond I]$*: Let* Γ *be a structure of type* A*. We can extend* Γ *at the root with the unary structure-building operator* $\blacklozenge$ *to obtain a structure of type* $\Diamond A$*.*

$$\frac{\Gamma \vdash A}{\blacklozenge(\Gamma) \vdash \Diamond A}\ [\Diamond I]$$

Axiom

Starting with the base case, the smallest possible derivation should be considered. This is, naturally, the derivation consisting of only one sequent that derives itself.

As there are no premises, the top part of the rule is empty. Therefore the line above the sequent is usually omitted. As it is clear from the sequent that it is an axiom, the label is sometimes also omitted. In a derivation, the axioms will form the leaves, as we will see later in the examples.

For the connectives, as stated earlier, both an introduction and elimination rule have to defined. The introduction rule for an operator introduces that operator in the conclusion while combining the premises. The elimination rule for an operator eliminates that operator in the conclusion while combining the premises. In this section we'll first present the elimination rules for the binary connectives, after which the introduction rules will be presented. In the last part of this section the unary connective's elimination and introduction rules will be presented.

Elimination rules for / and \

The elimination rules of the slashes are the most intuitive, as one type is incomplete to its right or left, and the other type completes the first type. Note that there is no slash in the conclusion and it is thus eliminated!

Elimination rule for •

The elimination rule for the product operator, •, involves substitution of a structure embedded in another structure.[2] The result of this rule is a structure that contains the substitute structure without a •, hence elimination of the •.

So far, the elimination rules have been discussed. As can be observed from the rules, they decompose a complex type in a premise into its subparts, e.g. A/B is decomposed into A and B. Now we turn to the rules that form a complex type out of its subparts, i.e. the introduction rules.

Introduction rules for / and \

If the introduction rules for / and \ are read bottom-up, we see in the conclusion that Δ is incomplete with respect to a type B on it's right (or left) and that in the premise, this type B is assumed to be present to form a type A. This kind of assumption is also known as *hypothetical reasoning*.

Introduction rule for •

From the introduction rule for the product operator, •, we see that the $\circ$ is actually the structural counterpart for the type-building operator, •.

[2] To represent an embedded structure the notation $\Gamma[\Delta]$ is used, where the structure Δ is contained within the Γ structure. Where Δ is positioned in Γ does not matter. It only matters that Δ is present somewhere within Γ.

With all derivation rules for binary connectives defined, we now turn to the unary connectives, $\Box$ and $\Diamond$. Remember that these connectives are 'feature' connectives, where the $\Box$ will unlock a constituent for this feature and a $\Diamond$ will lock it.

Derivation rules for $\Box$

In the derivation rules for $\Box$, we see that the feature of the $\Box$ is moved to either the structural domain, denoted with a $\blacklozenge$ or to the logical domain, denoted with a $\Box$ as we know it. When structural control postulates will be added to this logic later on we'll see the use of these operators in full effect. For now, we'll turn to the derivation rules of the $\Diamond$.

Derivation rules for $\Diamond$

In the introduction rule of the $\Diamond$, we see how the lock on the structures are removed if read bottom-up. In the elimination rule, we see how we can remove the lock top-down.

Examples

Now the derivation rules are established, they can be used to derive some example sentences. Note that these derivation rules are of a syntactical nature, i.e. they do not tell us anything about the meaning assembly that can be derived with them yet. Also note, that in these examples no hypothetical reasoning is used and therefore no introduction rules are applied.[3]

$$\dfrac{\dfrac{\textbf{John}}{np \vdash np} \quad \dfrac{\textbf{slept}}{np\backslash s \vdash np\backslash s}}{np \circ np\backslash s \vdash s}\,[\backslash E]$$

$$\dfrac{\dfrac{\textbf{they}}{\Box_{pl} np \vdash \Box_{pl} np} \quad \dfrac{\textbf{sing}}{\Box_{pl} np\backslash s \vdash \Box_{pl} np\backslash s}}{\Box_{pl} np \circ \Box_{pl} np\backslash s \vdash s}\,[\backslash E]$$

$$\dfrac{\dfrac{\dfrac{\textbf{the}}{np/n \vdash np/n} \quad \dfrac{\textbf{man}}{n \vdash n}}{np/n \circ n \vdash np}\,[/E] \quad \dfrac{\textbf{ate}}{(np\backslash s)/np \vdash (np\backslash s)/np} \quad \dfrac{\dfrac{\textbf{the}}{np/n \vdash np/n} \quad \dfrac{\textbf{apple}}{n \vdash n}}{np/n \circ n \vdash np}\,[/E]}{(np\backslash s)/np \circ (np/n \circ n) \vdash np\backslash s}\,[/E]}{(np/n \circ n) \circ ((np\backslash s)/np \circ (np/n \circ n)) \vdash s}\,[\backslash E]$$

2.1.3 Derivation rules with semantics

One of the benefits of typelogical grammar is that semantic interpretation occurs simultaneously with the syntactical derivation. So far in this section, we have established all derivation rules for the operators considering the syntactical aspect. How the derivation rules assemble a semantic term has been omitted until now. In this part, we'll present all derivation rules again, but with

[3] In the example derivations the word of the lexicon that corresponds to the axiom is written above the axiom to clarify what has been derived so far.

the semantical values *decorated* over them.

For the *syntax-semantics interface*, we'll turn to the Curry-Howard correspondence between derivations and meaning assembly. The Curry-Howard correspondence is a link between logic and computation and provides us with the means to connect the syntax (logic) of a sentence to its meaning (computation). It is a derivational view on semantics, and thus fits perfectly within typelogical grammar. Each derivation or inference step is associated with a composition of semantic terms. When a derivation is complete, the final conclusion will return a semantic term that represents the meaning of a sentence. Introduced by van Benthem [1988] into the categorial grammars, the Fregean principle of compositionality is realized, i.e. the principle that 'the meaning of a complex expression is a function of the meaning of its parts and the rules that put them together'.

The language we will use for the semantics of natural language phrases will be the typed lambda-calculus. In order for us to use that language, we'll first need a notion of how to translate the syntactic types we have used so far to semantic types that can be used by the typed lambda-calculus. After that has been established, the typed lambda-calculus will be discussed in more detail and the derivation rules can be presented. The lexicon we presented before will also be extended with semantics for illustrational purposes.

Semantic types

Similar to the syntactic types, the set of semantic types consists of basic types and type-forming operations to construct complex types. The basic types will, as usual in formal semantics, be e denoting entities and t denoting truth-values. Note that this set of basic types is for illustration purposes and is by no means restricted to these two types! For the interpretation of questions, different types could be added to the set of basic types to ensure a distinction between declaratives and questions.

Definition 11 (Semantic type language) *Let $Atom_{sem}$ be the set $\{e, t\}$. The set of semantic types, Typ_{sem}, is the closure of $Atom_{sem}$ under the type-forming operations $\longrightarrow$ (function) and $\circ$ (product).*

$$Typ_{sem} ::= Atom_{sem} \mid Typ_{sem} \longrightarrow Typ_{sem} \mid Typ_{sem} \circ Typ_{sem}$$

For each type A in Typ_{sem}, we want to be able to refer to its domain of interpretation. For entities, we'll simply use D_e consisting of every entity in the model. For truth-values, we'll use D_t being the set $\{0, 1\}$. For more complex types, we'll follow an inductive approach.

Definition 12 (Denotation domains)

$$\begin{array}{lll} D_e & = E & (= \textit{the set of individuals}) \\ D_t & = \{0,1\} & (= \textit{the set of truth values}) \\ D_{A \longrightarrow B} & = D_B^{D_A} & (= \textit{the set of functions from } D_A \textit{ to } D_B) \\ D_{A \circ B} & = D_A \times D_B & (= \textit{the Cartesian product of } D_A \textit{ and } D_B) \end{array}$$

Now we have established the semantic type system, we can now move on to define a mapping from syntactic types to semantic types. Via the semantic type, each syntactic type will also have a corresponding denotation domain.

Definition 13 (Mapping of syntactic types to semantic types) *We define a function sem that maps each syntactic type to a semantic type. The base cases of the definition apply to the basic syntactic types:*

$$\begin{array}{ll} sem(np) & = e \\ sem(s) & = t \\ sem(n) & = e \longrightarrow t \end{array}$$

Recursive cases:

$$\begin{array}{rcccl} sem(A/B) & = & sem(B\backslash A) & = & sem(B) \longrightarrow sem(A) \\ & & sem(A \bullet B) & = & sem(A) \circ sem(B) \\ sem(\diamondsuit A) & = & sem(\square A) & = & sem(A) \end{array}$$

Notice that the syntactic type n is mapped to the semantic type $e \longrightarrow t$, and that the directionality of the slashes is important in the syntactic dimension, but not in the semantic dimension. Moreover, we see that a syntactic type containing a slash, such as A/B, is interpreted as a *function* from B to A, where B is the argument. Also notice that the unary connectives have no semantic value at all. They are purely used in the syntactic dimension.

Typed lambda-calculus

It is time to turn to the typed lambda-calculus. We will find that it will provide us with a nice, intuitive way of looking at natural language. Most importantly, it will take a functional approach, where a part of the sentence is regarded to be a function, and a different part regarded as the argument to that function.

Definition 14 (λ-terms) *Let C^A be a set of constants of type A and V^A a (denumerably infinite) set of variables of type A. The full set of typed lambda terms of type A, $\mathcal{T}^A$ is defined by the following grammar:*

$$\begin{array}{ll} \mathcal{T}^A & ::= \mathcal{V}^A \mid \mathcal{C}^A \mid (\mathcal{T}^{B \longrightarrow A}\ \mathcal{T}^B) \mid (\pi^1\ \mathcal{T}^{A \circ B}) \mid (\pi^2\ \mathcal{T}^{B \circ A}) \\ \mathcal{T}^{A \longrightarrow B} & ::= \lambda \mathcal{V}^A.\ \mathcal{T}^B \\ \mathcal{T}^{A \circ B} & ::= \langle \mathcal{T}^A, \mathcal{T}^B \rangle \end{array}$$

Definition 15 (Semantic value of λ-terms) *Let I (the interpretation function) and g (the assignment function) be functions associating constants, respectively variables, of type A with semantic objects in D_A. We define the semantic value of a term relative to such an interpretation and assignment function as follows:*

$$\begin{array}{ll} [\![x_A]\!]^{I,g} & = g(x) \text{ (where } x \text{ is a variable of type } A) \\ [\![t_A]\!]^{I,g} & = I(t) \text{ (where } t \text{ is a constant of type } A) \\ [\![(\phi\ \psi)_B]\!]^{I,g} & = [\![\phi_{A \longrightarrow B}]\!]^{I,g}\ ([\![\psi_A]\!]^{I,g}) \\ [\![(\lambda x.\phi)_{A \longrightarrow B}]\!]^{I,g} & = h_{A \to B} \text{ such that } \forall m \in D_A, h(m) = [\![\phi_B]\!]_I^{g[x:=m]} \\ [\![\langle \phi, \psi \rangle_{A \circ B}]\!]^{I,g} & = \langle [\![\phi_A]\!]^{I,g}, [\![\psi_B]\!]^{I,g} \rangle \\ [\![(\pi^1 \langle \phi, \psi \rangle)_{A \circ B}]\!]^{I,g} & = [\![\phi_A]\!]^{I,g} \\ [\![(\pi^2 \langle \phi, \psi \rangle)_{A \circ B}]\!]^{I,g} & = [\![\psi_B]\!]^{I,g} \end{array}$$

In this definition, the interpretation function, I, ensures that all constants in the logical domain have a corresponding value in the semantic domain. The assignment function simply assigns values to variables, for instance, $g[x := m]$ assigns the value m to the variable x.

Along with the semantic value of λ-terms, we'll find that different λ-terms have the same semantic value, meaning that these terms can be reduced to a simpler, or *normal*, form. In the following definition, a few standard reduction rules are presented. They are used to rewrite a complex term, the *redex*, to a simpler term, the *contractum*.

Definition 16 (Reduction rules) *β and η reductions for function and product types in lambda term equations:*

*β-**reduction**: When a semantic (sub)term is of the form $(\lambda x.t[x]\ u)$ and the semantic type of u matches the semantic type of x, substitute for each occurrence of variable x in term t the term u (schematically represented as $t[u/x]$).*

$$(\lambda x^A.t^B\ u^A) \rightsquigarrow_\beta t^B[u/x]$$

*π-**conversion**: When a semantic (sub)term is of the form $(\pi^1\langle t, u\rangle)$ or $(\pi^2\langle t, u\rangle)$, the term at the position which is indicated by the projector π becomes the reduced term:*

$$(\pi^1\langle t, u\rangle^{A\circ B}) \rightsquigarrow_\pi t^A \qquad (\pi^2\langle t, u\rangle^{A\circ B}) \rightsquigarrow_\pi u^B$$

*η-**conversion**: When a semantic term is of the form $(\lambda x.\ t\ x)$, a function type, or of the form $(\pi^i\langle t, u\rangle)$, a product type, η-reduction yields the following reduced terms:*

$$\lambda x^A.(t\ x)^B \rightsquigarrow_\eta t^{A\to B} \qquad \langle(\pi^1\ t^{A\circ B}), (\pi^2\ t^{A\circ B})\rangle \rightsquigarrow_\eta t^{A\circ B}$$

Now we have the typed lambda-calculus with types corresponding with syntactic types established thanks to the Curry-Howard correspondence, we are able to decorate our lexicon with semantic values. In the following we present the lexicon we presented before that only contained the syntactic types, but now we'll omit the types containing unary connectives, as they have no semantic value. For a lexicon containing semantics, we will maintain the following lexical description:

$$\text{Structure} \vdash \textbf{Term} : \mathit{Type}$$

In the remainder of this thesis we'll sometimes omit the types for the semantic variables, constants and functions, as they can be computed from the semantic mapping $sem(\cdot)$. For some logical constants we'll have standard type definitions, such as the quantifier symbols, $\forall$ and $\exists$, of type $(e \to t) \to t$ and boolean operations of type $t \to (t \to t)$, etc. The definite description operator, ι, is of type $(e \to t) \to e$, and looks for a *unique* element that satisfies the requirement of a given predicate, for instance, *'the man'* refers to a specific man and no other man. By convention, we'll use lowercase letters at the end of the alphabet for variables and capital letters for predicate variables.

John	$\vdash \mathbf{j}$	$: np$
slept	$\vdash \lambda x.(\mathbf{sleep}\ x)$	$: np\backslash s$
ate	$\vdash \lambda x.\lambda y.((\mathbf{eat}\ x)\ y)$	$: (np\backslash s)/np$

the	$\vdash \lambda Q.\iota z.(Q\ z)$	$: np/n$
man	$\vdash \lambda x.(\mathbf{man}\ x)$	$: n$
apple	$\vdash \lambda x.(\mathbf{apple}\ x)$	$: n$
somebody	$\vdash \lambda P.(\exists x.(P\ x))$	$: s/(np\backslash s)$

And finally, it's time to decorate the derivation rules with semantics values. In Figure 2, all derivation rules are repeated, but with semantic values corresponding to the rules. Once again, we omitted the derivation rules for the unary connectives as the semantic term is not changed by introduction or elimination of a unary connective.

Definition 17 (Term labeling of the derivation rules) *$[Ax]$: The term labeling of axioms is relatively straightforward and simply assigns a variable to a type.*

$$\frac{}{x : A \vdash x : A}\ Ax$$

$[\backslash E]$ and $[/E]$: The elimination rules of the two slashes ($/$ and $\backslash$) correspond with function application. Combining two expressions yields a term where term u, the function, is applied to term v, the argument.

$$\frac{\Delta \vdash v : B \quad \Gamma \vdash u : B\backslash A}{\Delta \circ \Gamma \vdash (u\ v) : A}\ [\backslash E] \qquad \frac{\Delta \vdash u : A/B \quad \Gamma \vdash v : B}{\Delta \circ \Gamma \vdash (u\ v) : A}\ [/E]$$

$[\backslash I]$ and $[/I]$: The introduction rules correspond with lambda abstraction. After abstracting the hypothesis from the antecedent structure, the term variable x assumed for the hypothesis is bound by a lambda operator.

$$\frac{x : B \circ \Delta \vdash u : A}{\Delta \vdash \lambda x.u : B\backslash A}\ [\backslash I] \qquad \frac{\Delta \circ x : B \vdash u : A}{\Delta \vdash \lambda x.u : A/B}\ [/I]$$

$[\bullet I]$ and $[\bullet E]$: The introduction rule for $\bullet$ corresponds to pairing, while the elimination rule associates with the projection of a term (u) with the already formed expression (t) by substituting the variables in that term.

$$\frac{\Delta \vdash t : A \quad \Gamma \vdash u : B}{\Delta \circ \Gamma \vdash \langle t, u \rangle : A \bullet B}\ [\bullet I] \qquad \frac{\Delta \vdash u : A \bullet B \quad \Gamma[x : A \circ y : B] \vdash t : C}{\Gamma[\Delta] \vdash t[\pi^1 u/x, \pi^2 u/y] : C}\ [\bullet E]$$

Examples with semantic decoration

As example derivations with semantics, we'll decorate semantic terms over the example derivations we have seen before for *"John slept"*, *"They sing"* and *"The man ate the apple"*. We omitted the derivation including unary connectives. Due to lack of space we sometimes write the syntactic types and semantic terms beneath each other.

$$\frac{\dfrac{\textbf{John}}{np \vdash \mathbf{j} : np} \quad \dfrac{\textbf{slept}}{np\backslash s \vdash \lambda x.(\mathbf{sleep}\ \mathbf{x}) : np\backslash s}}{np \circ np\backslash s \vdash \mathbf{sleep}\ \mathbf{j} : s}\ [\backslash E]$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\textbf{the}}{np/n \vdash \lambda Q.\iota z.(Q\ z)} \quad \dfrac{\textbf{man}}{n \vdash \lambda x.(\textbf{apple}\ x)}}{\dfrac{:np/n \qquad :n}{np/n \circ n \vdash \iota z.(\textbf{man}\ z)}\ [/E]} \quad \dfrac{\dfrac{\textbf{ate}}{(np\backslash s)/np \vdash \lambda x.\lambda y.((\textbf{eat}\ x)\ y)} \quad \dfrac{\dfrac{\textbf{the}}{np/n \vdash \lambda Q.\iota z.(Q\ z)} \quad \dfrac{\textbf{apple}}{n \vdash \lambda x.(\textbf{apple}\ x)}}{\dfrac{:np/n \qquad :n}{np/n \circ n \vdash \iota z.(\textbf{apple}\ z)}\ [/E]}}{\dfrac{:(np\backslash s)/np \qquad :np}{(np\backslash s)/np \circ (np/n \circ n) \vdash \lambda y.((\textbf{eat}\ (\iota z.(\textbf{apple}\ z)))\ y)}\ [/E]}}{\dfrac{:np \qquad :np\backslash s}{(np/n \circ n) \circ ((np\backslash s)/np \circ (np/n \circ n)) \vdash (\textbf{eat}\ (\iota z.(\textbf{apple}\ z)))\ (\iota w.(\textbf{man}\ w)) : s}\ [\backslash E]}$$

The semantic term for the sentence *"The man ate the apple"* can intuitively be read as: There is one unique man that ate one unique apple. In this reading, one should remember that the word unique means that there exists an entity such that there is no other entity that satisfies the given predicate, e.g. the only man in the room ate the only apple.

2.2 Conclusion

In this chapter we have presented the typelogical grammar that will be used in the remainder of this thesis. We observe that it uses its derivation rules to establish derivations for sentences, based on the types that are assigned to the individual words via the lexicon. However, one of the challenges this thesis aims to overcome is that the typelogical grammar presented requires sentences to be complete before a derivation can be established, as in the following example:

$$\dfrac{\dfrac{\dfrac{\textbf{the}}{np/n \vdash np/n} \quad \dfrac{\textbf{man}}{n \vdash n}}{np/n \circ n \vdash np}\ [/E] \quad \dfrac{\dfrac{\textbf{ate}}{(np\backslash s)/np \vdash (np\backslash s)/np} \quad \dfrac{\dfrac{\textbf{the}}{np/n \vdash np/n} \quad \dfrac{\textbf{apple}}{n \vdash n}}{np/n \circ n \vdash np}\ [/E]}{(np\backslash s)/np \circ (np/n \circ n) \vdash np\backslash s}\ [\backslash E]}{(np/n \circ n) \circ ((np\backslash s)/np \circ (np/n \circ n)) \vdash s}$$

The sentence *"The man ate the apple."* is not derived incrementally, as the phrase *"the apple"* has to be derived, before it can be combined with the word *"ate"*. In this thesis, incrementality will be understood as word-by-word, from left-to-right interpretation. In the example, where the derived structure according to the typelogical grammar is ((*the man*) (*ate* (*the apple*)), we are looking for an incremental structure of the form ((((*the man*) *ate*) *the*) *apple*) . This means that, for example, *"the man ate"* will have to derived first, before it can be combined with *"the"*.

So, the typelogical grammar we have presented is not inherently incremental. It can not iterate through the words incrementally and connect every next word to the preceding phrase. Later in this thesis, we will see that this is possible if the typelogical grammar is extended with a system that accommodates incrementality. To guarantee incrementality, associativity (or rebracketing) is necessary, which is lacking in the grammar we have presented at this point. Associativity is an extra derivation rule that could be added to the typelogical grammar, namely the following:[4]

[4]Note that the double horizontal line is simply a manner of stating this rule can be applied both ways (similar to a bi-implication in logics).

$$\frac{\Gamma[(\Delta_1 \circ \Delta_2) \circ \Delta_3] \vdash C}{\Gamma[\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash C}\ [Ass]$$

Associativity can be applied to any substructure of a structure Γ, and adding it to the typelogical grammar has consequences for the expressivity of the grammar. Adding a rule such as associativity entails that sentences that were underivable without this rule, are possibly derivable with this rule. Adding associativity therefore enlarges the set of derivable sentences, and enlarges the expressivity of the grammar. The typelogical grammar we have presented does not assume associativity and is therefore called a grammar of **NL**. A grammar that does assume associativity is called a grammar of **L**, which is the original grammar due to Lambek [1958].

The difference between **L** and **NL** is of a structural nature. They are called modalities, and encapsulate the same set of derivation rules. However, **L** has access to the associativity postulate to derive sentences, whereas **NL** does not. This has consequences in the expressivity of the grammar that is an instance of it. Naturally, a grammar of **L** is able to derive more sentences, since the associativity rule is present. It is therefore more flexible than a grammar of **NL**. Consequently, a grammar of **NL** is a more restricted grammar, as it does not possess such a structural postulate, and is not able to derive the same set of sentences as a grammar of **L**. The more restricted grammar of **NL** is therefore less expressive than the more flexible grammar of **L**. On the same level as expressivity as **L** there exists **NLP**, which is the modality **NL** extended with a structural postulate for commutativity (reordering). We will see a variant of such a postulate later in this thesis. But even more expressive is the modality **LP**, which is the modality **NLP** extended with the associativity postulate, or the modality **L** extended with the commutativity postulate. The expressivity of a grammar can be extended with a multimodal approach, where a grammar might have operators from two or more modalities. Structural postulates can be defined on structures containing these operators in a specific order, and the grammar can therefore enjoy the flexibility of, for example, associativity of **L**, but still be restricted as **NL**. Later in this thesis we will use the unary operators $\blacklozenge$ and $\Box$ to take such an approach.

Associativity allows us to rebracket a sentence, meaning that we can rebracket a sentence in such a way that it has a bracketing desired for incrementality. Later in this thesis, we add a structural variant of associativity to the grammar to make sure that we do not derive more than necessary.

Another subject that needs a short mention is that the lambda-calculus presented in this chapter is a typed variant. The future utterances of the speaker may be unknown for incremental interpretation, but when incrementally interpreting the words that have been uttered, the addressee also knows what kind of word has to be the next one. At least, that is if the addressee is hearing his/her native language, as an in-depth knowledge or intuition is needed to be able to reason about that language. Anyway, later in this thesis we will see types such as s/np, for example. This type already indicates that the next word has got to be of type np. This may be reflected in a semantic term as well as $sem(np)$ translates to e. In this case it is known that the argument to the type s/np in its semantic dimension has got to be of type e. If a different argument is provided to the semantic term, it will not be able to beta-reduce until a proposition has been reached. That means that the sentence given is ungrammatical. In this thesis we will omit the types of variables and functions whenever a semantic term is given. Moreover, every sentence that will be incrementally interpreted in the examples of this thesis will be a grammatical sentence, so there is no need for typing the semantic terms. But do remember that the types exist implicitly.

The typelogical grammar we have presented in this chapter is, despite these shortcomings, our starting point in the quest towards incremental interpretation. It will lend itself very well for it, as the types of wh-phrases have already been defined before (Chapter 3), and a system for incrementality in **L** has also been defined before (Chapter 5.2). We will find that it is possible to model incremental interpretation with this typelogical grammar under the addition of a structural variant of associativity (and commutativity, but more on that later) and adapting the system for incremental interpretation in **L** to accommodate the structural variant of associativity instead of the regular variant.

Chapter 3

Wh-questions

3.1 Typology of questions

To provide a picture of what questions really are, a taxonomy of questions by Kearsley [1976] will be discussed. He proposed a taxonomic scheme for questions to illustrate how questions can be classified based on their properties. The taxonomy is shown in Figure 1. Kearsley first claims that the goal of any question is always to elicit a response from a party that receives the question (addressee). He then goes on to divide questions into questions that are verbal and non-verbal. Non-verbal questions may be raising an eyebrow or shrug of the hands (overt) or an internally directed question that we answer ourselves (covert), while verbal questions are actually uttered to an addressee.

Verbal questions in themselves are subdivided into direct and indirect questions. Indirect questions are declaratives that contain an embedded interrogative, and are usually used to announce some kind of uncertainty of which the questioner wants it to be resolved, such as *"I wonder how tall Bill is."*. They are not questions in the syntactic sense, i.e. missing a question mark, but are questions nonetheless as they serve to elicit a response. Not every embedded clause construction is a question, though. For example, the sentence *"I know how tall Bill is."* contains an embedded clause, but the phrase *"know"* indicates a declarative clause instead of an interrogative clause. Therefore, this sentence is not an indirect question.

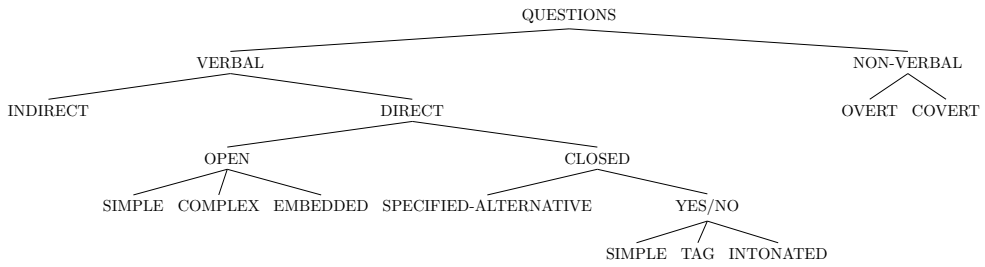


Figure 3.1: Taxonomy of questions by Kearsley [1976]

Direct questions, as opposed to indirect questions, are questions in the syntactic sense, as they end with a question mark. They can be divided into open and closed questions, where closed questions specify the possible answers to the questions either explicitly or implicitly. *"Is that dog dead?"* is an example of closed questions as it implicitly specifies the answers *'Yes'* and *'No'*. An example of explicitly specified answers in a closed question might be *"Is Amsterdam, Rotterdam or Utrecht the biggest city of the Netherlands?"* as the possible answers to the question are already stated. The subdivision of yes/no-questions is simply the way a yes/no-question is formulated. Simple yes/no-questions have an initial auxiliary verb, such as *"Is that dog dead?"*. Tagged yes/no-questions have inverted auxiliary verb at the end, such as *"That dog is dead, isn't it?"*. And intonated yes/no-questions are declaratives that are intonated in such a way that it forms a question after all, such as *"That dog is dead?"*.

However, wh-questions are not contained in the class of closed questions, as they do not explicitly or implicitly specify the possible answers to the questions. Kearsley states that open questions are always formed by use of a wh-phrase such as *'who'*, *'what'*, *'when'*, *'where'*, *'why'*, *'which'*, *'whose'* and *'how'*, which is stressed when uttered, and are therefore called wh-questions. In their place, they can be subdivided into a number of categories. Note that the subdivision of wh-questions of Vermaat [2006] will be adopted and we'll leave the taxonomy of Kearsley for the reason that Vermaat's subdivision is more specific with respect to wh-questions than Kearsley's. The sub-categories of wh-questions, according to Vermaat, are the following:

- Local questions
- Non-local questions
- Questions with island constraints
- Multiple wh-questions

Before each sub-category of wh-questions is discussed, it is important to mention here that there exist two kinds of wh-phrases, namely ex-situ and in-situ wh-phrases. Ex-situ wh-phrases are not interpreted in place, while in-situ wh-phrases are. In English, ex-situ wh-phrases are always at the start of the sentence, while in-situ are not. Consequently that means that in-situ wh-phrases can not exist in a wh-question without an ex-situ wh-phrase in that same question. Vermaat also makes this distinction between wh-phrases to make sure that the different variants are interpreted correctly in typelogical grammars, as we will see later.

Local questions

Local questions are subdivided in two categories, direct and indirect questions. Direct questions are questions that show wh-fronting, which means that the wh-phrase is present at the beginning of the sentence.[1] For example:

- *Who saw John?*
- *Whom did John see?*
- *What did John give to Mary?*

[1] In this case, only argument wh-phrases, such as *'who'* and *'what'* are discussed. Adjunct wh-phrases, such as *'how'* and *'why'* are left out of scope in this thesis.

In these questions, the wh-phrase create a gap. This gap symbolizes the answer to the question, i.e. the answer to the question could be filled in at the location of the gap to create a proposition that has a truth-value. In the first question a subject-gap is created with *"Who"*. In the second question *"Whom"* creates a direct object-gap. And in the third question an indirect object-gap is created by the word *"What"*. When a non-subject-gap is created, do-support is needed, as shown in the second and third question. The English language has this property, but the Dutch language, for example, does not. In Dutch, the first and second question will translate to the same sentence *"Wie zag John?"*. This is a source of ambiguity in Dutch.[2]

Indirect local questions do not have wh-fronting, but contain a subordinate or embedded clause of which the first word will be a wh-phrase. These questions contain verbs such as *'wonder'* and *'forget'*. Note that this kind of wh-question does not end with a question mark, hence the 'indirect'. For example:

- *John wonders whether Mary is tall.*
- *Bill forgot who saw a bird.*
- *Richard wonders what Mary saw.*

Indirect local questions differ from direct local questions, since they do not need do-support to form a question, as in the third question. Note that these questions are indirect and are thus not contained in the category that Kearsley claimed wh-questions to be. As they are declaratives that contain an embedded question and do not end with a question mark, they will be left out of the scope of this thesis.

Non-local questions

Non-local questions are characterized by having a wh-phrase at the start of the sentence, but the associated gap of that wh-phrase appears in a subordinate or embedded clause. They need do-support in the main clause and any number of embedded clauses may follow that main clause including bridging verbs such as *'believe'*, *'think'* and *'say'*. The complementizer *'that'*, which normally introduces an embedded clause must be omitted when the gap of the question takes a subject-position, whereas it is optional when it takes a non-subject-position. The following sentences demonstrate this:

- *Who did Bill think (*that) Mary saw?*
- *What did Bill believe (that) Mary saw?*
- *What did Bill believe (that) John said (that) Sue claimed (that) Mary saw?*

Questions with island constraints

Questions with island constraints are questions that contain syntactic islands off which phrases can't get off, much like a person can't get of an island without use of a boat or bridge. When a question contains such an island, wh-questions are harder to form, since if the gap of the

[2]This is due to the fact that Dutch is a so-called V2 language, meaning that the main verb is always the second part of the sentence. English is an exception to this, and therefore requires do-support when a non-subject gap is created.

wh-questions is outside of the scope of the island, they are syntactically incorrect. For example, look at:

- * *Which bird did John wonder [whether Mary wanted to see]?*
- * *Which bird did John wonder [who wanted to see]?*

Both these questions are subject to a wh-island (the bracketed phrases, starting with a wh-phrase) and the gap in question is about the bird, and thus outside of the scope of the island. Wh-islands are not the only kind of island constraints. There exist adjunct island constraints, complex noun phrase constraints and coordinate structure constraints as well, but as island constraints make questions syntactically complex, this kind of wh-question will be left out of the scope of the thesis. For more information on island constraints, the reader is referred to notes by Munn [2016].

Multiple wh-questions

Lastly, multiple wh-questions contain two or more wh-phrases. According to Cheng [2003], in English, one of these phrases is wh-fronted and the others stay in-situ, i.e. do not undergo overt movement of the wh-phrase (*wh-movement*). Rudin [1988] states that other languages have different ways of dealing with multiple wh-phrases in a question, but in this thesis we'll focus on the English multiple wh-questions mainly. Examples in English are, among others:

- *Who did John persuade to read what?*
- *What did Mary buy when?*

As multiple wh-questions also contain multiple gaps, the incremental interpretation of these questions is very interesting. What should a correct answer to such a question exist of? However, due to time constraints, multiple wh-questions have been left out of the scope of this thesis.

3.2 Wh-questions in typelogical grammars

Wh-phrases such as *'who'*, *'what'*, *'where'*, *'why'* and *'which'* all behave in more or less the same fashion, in the sense that they all create a gap hypothesis in a wh-question. Wh-phrases can differ in structural behaviour, i.e. in-situ vs. ex-situ, meaning that they create the gap hypothesis at a different location in the wh-question. Semantically, wh-phrases will all have different meaning assemblies. For instance, of the word *'who'* we know that it refers to a person, while the word *'what'* might refer to either an object or event. To interpret wh-phrases incrementally within typelogical grammar by Moortgat [1997], they need to be classified as belonging to a certain category indicating their behaviour within the wh-question they are embedded in.

Vermaat [2006] proposed a q-type schema, based on Moortgat [1996], that contains three variables: $\mathrm{WH}(A, B, C)$. Moortgat [1996] used his q-type schema to account for in-situ binding of generalized quantifiers, which share properties with wh-phrases. In a sentence, generalized quantifiers occupy the same position that a noun phrase would, but semantically they take scope over the clause in which they are embedded. For example, in the sentence *"Bill sees something"* with semantic term $\exists x.\ (\mathbf{see\ bill}\ x)$, which is composed out of the terms $\lambda P.\exists x.(P\ x)$ for *"something"* and $\lambda x.(\mathbf{see\ bill}\ x)$ for *"Bill sees"*, we observe that the quantifier $\exists$ takes scope over the phrase

"Bill sees". Wh-phrases, on the other hand, create a gap hypothesis at the location of the word that was questioned, but semantically also take scope over the clause in which the gap is embedded. For example, in the sentence *"Bill saw John"* the gap can be converted to a wh-question by questioning the subject. The result will be *"Who t_1 saw John?"*, where t_1 indicates the location of the gap. The corresponding semantics to that question might be, for example, given by the term $\lambda x.(\mathbf{see}\ x)\ \mathbf{bill}$, where the λ binds the answer to the question.

The semantics of a question is taken to be a lambda-term that still requires an argument. That argument will be the variable that binds the answer to the question. Such a semantic term and the answer is as we saw above. Together the question and answer compose a proposition that can be evaluated to be true or false. As the answer to a question is represented by a lambda-abstraction in the semantic term of the question, it needs to be accounted for by the wh-type schema. In the final part of this section we'll see how the wh-type schema by Vermaat accounts for that lambda-abstraction of the answer.

In the schema, A will be the type of the gap hypothesis in the body of the question created by the wh-phrase, B the type of the body of the wh-question and C the type of the expression after merging the wh-phrase with the body. The schema can be used to restrict the hypothesis and body to specific types (A and B respectively) and determine what the result type after merging the wh-phrase into the body is (C). Vermaat proposed the following rule to illustrate the merging of wh-phrase and body in Gentzen's sequent-style presentation:

$$\frac{\Gamma \vdash \mathrm{WH}(A, B, C) \quad \Delta[A] \vdash B}{\Delta[\Gamma] \vdash C}\ [\mathrm{WH}]$$

As we can read from the WH-rule, a substructure of type A of a structure Δ, that constitutes the body of type B, is substituted by a structure Γ that is the wh-phrase. The following derivation of the question *"Who saw Bill?"* will make this clearer:[3]

$$\dfrac{\dfrac{\mathbf{who}}{\mathbf{who} \vdash \mathrm{WH}(np, s, wh)} \qquad \dfrac{[np \vdash np]^1 \qquad \dfrac{\dfrac{\mathbf{saw}}{(np\backslash s)/np \vdash (np\backslash s)/np} \quad \dfrac{\mathbf{bill}}{np \vdash np}\ /E}{\mathbf{saw} \circ \mathbf{bill} \vdash np\backslash s}\ \backslash E}{np \circ (\mathbf{saw} \circ \mathbf{bill}) \vdash s}\ [\mathrm{WH}]}{\mathbf{who} \circ (\mathbf{saw} \circ \mathbf{bill}) \vdash wh}$$

Here, the wh-phrase *'who'* is typed as the wh-type schema with its variables instantiated with np, s and wh respectively. The type wh is used by Vermaat to indicate a wh-question. The top part of the derivation is fairly straightforward as the phrases *'saw'*, *'bill'* and the hypothesis of type np can be combined easily with the elimination rules of the slashes. Then, we see that both the types of the gap hypothesis and the body in the wh-type schema, $\mathrm{WH}(np, s, wh)$, are in accordance with the until now derived sequent, $np \circ (\mathbf{saw} \circ \mathbf{bill}) \vdash s$, which is why the wh-phrase can be merged with the sentence. The result is the phrase *"Who saw bill"* of type wh.

On the semantic side of the schema, Vermaat defines, according to the Curry-Howard correspondence, a function of type $(\overline{A} \to \overline{B}) \to \overline{C}$ to represent the schema $\mathrm{WH}(A, B, C)$, where $\overline{A}$ represents the type of the gap of the question, $\overline{B}$ the type of the body of the question, $\overline{C}$ the type that is the result of merging the body with the wh-phrase. The type of the semantics shows these steps as the first argument is a function of type $\overline{A} \to \overline{B}$ which represents the abstraction

[3] Once again, the words of the lexicon to which the axioms correspond are written above the axioms. Moreover, we use words in the left-hand side of the sequents instead of structures for clarification purposes. These words correspond to the type defined by the lexicon and are used as if they were structures.

of the gap hypothesis over the body and the application to form a type $\overline{C}$ represents the merger of the body and wh-phrase. The actual function of the wh-type schema is $(\omega\ \lambda x^{\overline{A}}.\ \text{BODY}^{\overline{B}})^{\overline{C}}$, where BODY is the semantic term of the body and ω the function that represents the merger of the body of which the hypothesis is abstracted and the wh-phrase. This semantic term can be used to represent any wh-type schema as A, B and C can be substituted by any type. The semantics of the wh-type schema can be easily incorporated in the WH-rule above as follows:

$$\frac{\Gamma \vdash \omega : \text{WH}(A, B, C) \quad \Delta[x : A] \vdash \text{BODY} : B}{\Delta[\Gamma] \vdash \omega\ \lambda x.\text{BODY} : C}\ [\text{WH}]$$

The ω-operator is simply a shorthand term for a lambda-term that takes the answer into account. However, we do not need to consider answers for now, so we'll leave the lambda-term that is the ω implicit. In short, it binds the gap hypothesis as a variable in the body of the question.

So far the general idea of the wh-type schema by Vermaat. There is, however, a difference in how wh-phrases are interpreted. Some wh-phrases create a hypothesis gap that does not occur at the position of the wh-phrase and others do. To account for those structural differences, Vermaat proposes three variants of the general wh-type schema, which we will discuss next.

3.2.1 Ex-situ wh-schema

When the wh-phrase shows wh-fronting, meaning they are at the initial position of the sentence (or clause), the hypothesis gap they create does not occur at that fronted position. Fronted wh-phrases should therefore be interpreted ex-situ (Latin: off-position). Vermaat proposes two variants of the wh-type schema to account for ex-situ wh-phrases. One of those schemata accounts for hypothesis gaps that occur on a right branch, and the other for hypothesis gaps that occur on a left branch. Together with structural displacement postulates to move the hypothesis gap to the correct position, Vermaat manages to account for ex-situ wh-phrase binding.

When the hypothesis gap occurs on a left branch of the body, the wh-phrase can be merged with the body with the following rule:

$$\frac{\Gamma \vdash \text{WH}^{l}_{ex}(A, B, C) \quad A \circ \Delta \vdash B}{\Gamma \circ \Delta \vdash C}\ [\text{WH}^{l}_{ex}]$$

Similar to the general WH-rule we saw before, the wh-phrase, denoted by Γ, replaces the gap hypothesis of type A in the body of type B. However, for the WH^{l}_{ex}-rule, it is important that the gap hypothesis is not a substructure of the body, as was earlier denoted by $\Delta[A]$, but exists on the same level as the rest of the body, as in $A \circ \Delta$. The wh-phrase is inserted at the left branch of the structure, as we can see from $\Gamma \circ \Delta$ in the conclusion of the rule.

When the hypothesis gap occurs on a right branch, the rule is analogous to the wh-ex-situ left rule:

$$\frac{\Gamma \vdash \text{WH}^{r}_{ex}(A, B, C) \quad \Delta \circ A \vdash B}{\Gamma \circ \Delta \vdash C}\ [\text{WH}^{r}_{ex}]$$

In this rule, we see that the hypothesis gap occurs on a right branch in the body, as in $\Delta \circ A$. The application of the wh-ex-situ right rule is, apart from the position of the hypothesis gap,

identical to the application of the wh-ex-situ left rule.

But what can we do if the hypothesis gap does not immediately appear on the left or right branch of the body, but is more deeply embedded into the structure? That's where the displacement postulates of Moortgat [1999] come in. Before we show the displacement postulates, it's useful to know that in order for these postulates to work correctly we need to decorate the hypothesis gap within the type schema with unary operators. The adding of unary operators to the hypothesis gap ensures that the displacement postulates are only applicable to the hypothesis gap and not to any other word in the structure of the body. Because of the logical rules the unary operators are restricted to (in particular the $\Diamond I$-rule), we have to add the combination of a $\Diamond$ and $\Box$ to the hypothesis gap. The wh-ex-situ left and right schemata will now look as follows:

$$\mathrm{WH}^{l}_{ex}(\Diamond\Box A, B, C) \qquad \mathrm{WH}^{r}_{ex}(\Diamond\Box A, B, C)$$

Where A, B and C still represent what they did before. The only difference is that the hypothesis gap is now obligated to be decorated by unary operators. And now, for the displacement postulates we see, when read top-down, that the structure decorated with $\blacklozenge$ is one level higher up in the conclusion than that it was in the premise:

Left displacement postulates

$$\frac{\Gamma[(\blacklozenge\Delta_1 \circ \Delta_2) \circ \Delta_3] \vdash C}{\Gamma[\blacklozenge\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash C}\ [Pl1] \qquad \frac{\Gamma[\Delta_2 \circ (\blacklozenge\Delta_1 \circ \Delta_3)] \vdash C}{\Gamma[\blacklozenge\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash C}\ [Pl2]$$

Right displacement postulates

$$\frac{\Gamma[\Delta_1 \circ (\Delta_2 \circ \blacklozenge\Delta_3)] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2) \circ \blacklozenge\Delta_3] \vdash C}\ [Pr1] \qquad \frac{\Gamma[(\Delta_1 \circ \blacklozenge\Delta_3) \circ \Delta_2] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2) \circ \blacklozenge\Delta_3] \vdash C}\ [Pr2]$$

The following example makes use of these displacement postulates and shows, when read top-down, that the hypothesis gap is moved towards the most upper level before the wh-phrase can be inserted. We make use of the following lexicon and derive the wh-question *"Whom did John see"*:

whom	:: $\mathrm{WH}^{r}_{ex}(\Diamond\Box np, q, wh)$
did	:: $q/(np \bullet inf)$
john	:: np
see	:: inf/np

$$\dfrac{\dfrac{}{\mathbf{whom} \vdash \mathrm{WH}^{r}_{ex}(\Diamond\Box np, q, wh)} \quad \dfrac{\dfrac{}{\Diamond\Box np \vdash \Diamond\Box np} \quad \dfrac{\dfrac{\dfrac{}{\mathbf{did} \vdash q/(np \bullet inf)} \quad \dfrac{\dfrac{}{\mathbf{john} \vdash np} \quad \dfrac{\dfrac{}{\mathbf{see} \vdash inf/np} \quad \dfrac{\dfrac{}{\Box np \vdash \Box np}}{\blacklozenge\Box np \vdash np}\,[\Box E]}{\mathbf{see} \circ \blacklozenge\Box np \vdash inf}\,[/E]}{\mathbf{john} \circ (\mathbf{see} \circ \blacklozenge\Box np) \vdash np \bullet inf}\,[\bullet I]}{\mathbf{did} \circ (\mathbf{john} \circ (\mathbf{see} \circ \blacklozenge\Box np)) \vdash q}\,[/E]}{\mathbf{did} \circ ((\mathbf{john} \circ \mathbf{see}) \circ \blacklozenge\Box np) \vdash q}\,[Pr1]}{(\mathbf{did} \circ (\mathbf{john} \circ \mathbf{see})) \circ \blacklozenge\Box np \vdash q}\,[Pr1]}{(\mathbf{did} \circ (\mathbf{john} \circ \mathbf{see})) \circ \Diamond\Box np \vdash q}\,[\Diamond E]}{\mathbf{whom} \circ (\mathbf{did} \circ (\mathbf{john} \circ \mathbf{see})) \vdash wh}\,[\mathrm{WH}^{r}_{ex}]$$

Note that in this derivation, the gap hypothesis is not in the same position as the position where the wh-phrase is inserted, hence 'ex-situ'.

The difference between the wh-ex-situ left schema, the wh-ex-situ right schema and the general schema from before is purely structural. Therefore the semantic term of both the wh-ex-situ schemata after merging is still: $(\omega\ \lambda x^{\overline{A}}.\ \mathrm{BODY}^{\overline{B}})^{\overline{C}}$

3.2.2 In-situ wh-schema

In contrast to the ex-situ wh-phrases, some wh-phrases create a hypothesis gap at the some position as where the wh-phrases occurs. These are called in-situ (Latin: in-position) wh-phrases. In-situ wh-phrases usually appear more deeply embedded within a question. For instance, in multiple wh-questions the fronted wh-phrase is interpreted ex-situ, but the other wh-phrases are interpreted in-situ. The rule Vermaat proposed for interpreting wh-phrases in-situ, is as follows:

$$\dfrac{\Gamma \vdash \mathrm{WH}_{in}(A, B, C) \quad \Delta[A] \vdash B}{\Delta[\Gamma] \vdash C}\,[\mathrm{WH}_{in}]$$

Note the similarity with the general WH-rule we discussed earlier. There are, however, some intricate differences in when the merger of the wh-phrase and body containing the hypothesis gap takes place and the composition of the semantic term.

In-situ wh-phrases behave a little different than their ex-situ variant. Since the ex-situ wh-phrases are always fronted in a question, they naturally take scope over the clause in which they are fronted. In-situ wh-phrases do not naturally take scope over the clause they are embedded in, but for the correct semantic representation of an in-situ wh-phrase, they should be able to take scope over that clause. As discussed before, in-situ wh-phrases bind a variable by a λ that represents the answer. The binding of the answer variable takes scope over the body of the clause the in-situ wh-phrase is embedded in to ensure that the eventual semantic term will still require an answer as argument.

To model this behaviour of in-situ wh-phrases, Vermaat shows that the hypothesis gap created by the wh-phrase once again has to move to the edge of the structure that forms the body of the wh-question, similar to the interpretation procedure of ex-situ wh-phrases. Ex-situ wh-phrases are then inserted at that edge position, but in-situ wh-phrases should be inserted in the original

position of the hypothesis gap (as it is in-position interpretation). To accommodate the insertion of the in-situ wh-phrase in the original position of the hypothesis gap, Vermaat decomposes the wh-in-situ type schema into two parts, the *scope marker* and the *gap*.

$$\mathrm{WH}_{in}(A,B,C) = \underset{\textit{(scope marker)}}{\Diamond\Box(C/B)} \bullet \underset{\textit{(gap)}}{A}$$

The scope marker is used to mark the position in the clause over which the wh-phrase will take scope. The position of the gap is used as the position where the wh-phrase will be inserted when merging the wh-phrase and the body. The scope marker and gap don't have to be at the same position and, consequently, the abstraction of the gap may take scope over the entire clause that the gap resides in.

$$\dfrac{\Gamma \vdash \Diamond\Box(C/B) \bullet A \qquad \dfrac{\Diamond\Box(C/B) \vdash \Diamond\Box(C/B) \qquad \dfrac{\dfrac{\dfrac{[\Box(C/B) \vdash \Box(C/B)]}{\blacklozenge\Box(C/B) \vdash C/B}\,[\Box E] \qquad \dfrac{[A \vdash A]}{\vdots}}{\blacklozenge\Box(C/B) \circ \Delta[A] \vdash C} \qquad \vdots\,[Pl1]}{\Delta[\blacklozenge\Box(C/B) \circ A] \vdash C}\,[\Diamond E]}{\Delta[\Diamond\Box(C/B) \circ A] \vdash C}\,[\bullet E]}{\Delta[\Gamma] \vdash C}$$

Note that the $\bullet$-elimination rule inserts the wh-phrase in the expression and both the scope marker and the hypothesis gap are hypotheses in the derivation. Also note that the derivation of the body Δ is left implicit in this derivation. So, although this is a useful decomposition of the wh-in-situ type schema, it also restricts the gap and scope marker to be next to each other before the wh-phrase can be inserted into the sentence. As the scope marker will always be inserted at the left edge, we need to move the scope marker to the position next to the gap. We can use exactly the same displacement postulates as before, but as those were used to extract the hypothesis out of deep embedding, we'll now use them in reverse to embed the scope marker more deeply with every step.

$$\dfrac{\Gamma[\blacklozenge\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash C}{\Gamma[(\blacklozenge\Delta_1 \circ \Delta_2) \circ \Delta_3] \vdash C}\,[Pl1] \qquad \dfrac{\Gamma[\blacklozenge\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash C}{\Gamma[\Delta_2 \circ (\blacklozenge\Delta_1 \circ \Delta_3)] \vdash C}\,[Pl2]$$

Note that we only need the left displacement postulates as the scope marker hypothesis will always be inserted at the left edge of the question body. Should the position of the scope marker hypothesis be different for any reason, the set of displacement postulates could easily be adjusted accordingly.

As to the semantical aspect of the wh-in-situ type schema, we want it to be the same lambda-term as for the ex-situ variants. The decomposed type of the schema can be used for that goal. The decomposed type consists of two parts, namely the scope marker, $\Diamond\Box(C/B)$, and the gap, A. According to the Curry-Howard correspondence these two types map to semantic types $\overline{B} \to \overline{C}$ and $\overline{A}$, respectively. The semantic type of the whole decomposed type schema will then be $(\overline{B} \to \overline{C}) \circ \overline{A}$. The lambda-term that represents the semantics of an in-situ wh-phrase must be

of this type. Vermaat uses the ω-operator of type $\overline{A} \to \overline{B}$ once again and a designated variable `X` to propose the following term:

$$\langle \lambda y^{\overline{B}}.\ (\omega\ \lambda \mathtt{X}.\ y)^{\overline{C}},\ \mathtt{X}^{\overline{A}} \rangle : \Diamond\Box(C/B) \bullet A$$

The use of the designated variable `X` ensures that the variable is affected simultaneously in both parts of the semantic term. This term together with the $\bullet$ elimination rule, will result in the semantic term that we already had for the ex-situ wh-phrases.

Example In English, the first wh-phrase will always occur fronted. That fronted wh-phrase is, as discussed above, always ex-situ. For English to make use of the in-situ variant of wh-phrases, there need to occur multiple wh-phrases in the same expression. In the following example, we'll therefore present the derivation of the multiple wh-question *"Who saw whom?"*. Due to the type-assignments for the subject wh-phrase *'who'* and the object wh-phrase *'whom'*, we'll find that the subject wh-phrase has to be inserted first, after which the object wh-phrase can be inserted. The following lexicon is used:[4]

$$\begin{array}{ll} \textbf{who} & :: \mathrm{WH}^{r}_{ex}(np, s, wh) \\ \textbf{saw} & :: (np\backslash s)/np \\ \textbf{whom} & :: \mathrm{WH}_{in}(np, wh, wh) \end{array}$$

$$\dfrac{\dfrac{\textbf{whom} \vdash \mathrm{WH}_{in}(np, wh, wh) \qquad \dfrac{\textbf{who} \vdash \mathrm{WH}^{r}_{ex}(np, s, wh) \qquad \dfrac{\dfrac{\textbf{saw} \vdash (np\backslash s)/np \qquad \overline{[np \vdash np]}}{\textbf{saw} \circ np \vdash np\backslash s}\,[\backslash E] \qquad \overline{[np \vdash np]}}{np \circ (\textbf{saw} \circ np) \vdash s}\,[/E]}{\textbf{who} \circ (\textbf{saw} \circ np) \vdash wh}\,[\mathrm{WH}^{l}_{ex}]}{\textbf{who} \circ (\textbf{saw} \circ \textbf{whom}) \vdash wh}\,[\mathrm{WH}_{in}]$$

3.3 Accounting for answers

Until now, we had a fairly vague variable ω in the semantic terms of both in-situ and ex-situ wh-phrases. The ω-operator encoded the abstraction of the hypothesis gap and the insertion of the wh-phrase into the body of the question. However, in this thesis, the meaning of questions will be taken to be lambda terms that still require an answer argument, and we have yet to account for the incorporation of answers into the semantic term of a wh-question. This section explains how the ω-operator is decomposed into a lambda term that will account for the answer to a question.

Before the ω-operator can be rewritten as a lambda term, it is necessary to adjust the wh-type schema itself to accomodate answers, i.e. the result type in the type schema after merging needs to be adjusted in such a way that it still requires an answer. The type $A/_?B$ operator will be used for this purpose, meaning that a new sentence of type B is expected to the right side as answer to compose a type A. To differentiate between composition on a sentential level and a dialogue level, a new composition operator $\circ_?$ is also introduced. The subscript ? is used for this

[4]Note that we do not use structural control operators in this examples as structural movement is not necessary for the gap hypotheses.

distinction.

Now that the answer to a question can be accounted for syntactically, the ω-operator can be rewritten as a lambda term. For illustration purposes, in this section we'll only rewrite the ω-operator for the wh-phrase *'who'* and *'which'*. Note that we have not taken into account that the wh-phrase *'who'* only refers to people and not, for instance, objects or events as the wh-phrase *'what'* does. The ω-operator for *'who'* is rewritten as the lambda term $\lambda P^{\overline{A}\to\overline{B}}.\lambda Q^{(\overline{A}\to\overline{B})\to\overline{B}}.(Q\ P)$. Recall that the semantic type of the wh-type schema is $(\overline{A}\to\overline{B})\to\overline{C}$, so the variable P in the lambda-term of ω will denote the body of the question in which the hypothesis gap of type $\overline{A}$ is abstracted. The wh-type definition for the wh-phrase *'who'* will now look as follows:

$$\mathbf{who} \vdash \lambda P^{\overline{A}\to\overline{B}}.\lambda Q^{(\overline{A}\to\overline{B})\to\overline{B}}.\ (Q\ P) : \mathrm{WH}(np, s, s/_?gq)$$

Note that gq is short for the type $s/(np\backslash s)$, which denotes a generalized quantifier. The variable Q will eventually account for the answer. For the question to have a meaningful semantic term, however, we do not necessarily need this variable to be instantiated. Nevertheless, we present an example derivation, where an answer is incorporated in the derivation, for the question-answer pair *"Who saw Mary? Nobody"*. Due to lack of space, the semantic and syntactic types of a phrase will occasionally be presented on a new line:

$$\dfrac{\dfrac{\mathbf{who} \vdash \lambda P.\lambda Q.(Q\ P)\quad : \mathrm{WH}(np, s, s/_?gq) \qquad \dfrac{x : np \vdash x : np \qquad \dfrac{\mathbf{saw} \vdash \mathbf{see} : (np\backslash s)/np) \quad \mathbf{mary} \vdash \mathbf{m} : np}{\mathbf{saw} \circ \mathbf{mary} \vdash \mathbf{see}\ \mathbf{m} : np\backslash s}\,[/E]}{x : np \circ (\mathbf{saw}\ \circ\ \mathbf{mary}) \vdash ((\mathbf{see}\ \mathbf{m})\ x) : s}\,[\backslash E]}{\mathbf{who}\ \circ\ (\mathbf{saw}\ \circ\ \mathbf{mary}) \vdash \lambda Q.(Q\ \lambda x.((\mathbf{see}\ \mathbf{m})\ x)}\,[\mathrm{WH}^l_{ex}] \qquad \mathbf{nobody} \vdash \lambda P.\neg\exists\lambda y.(P\ y)}{(\mathbf{who}\ \circ\ (\mathbf{saw}\ \circ\ \mathbf{mary}))\ \circ_?\ \mathbf{nobody} \vdash \lambda Q.(Q\ \lambda x.((\mathbf{see}\ \mathbf{m})\ x))\ (\lambda P.\neg\exists\lambda y.(P\ y)) : s}\,[/E]$$

$$: s/_?(s/(np\backslash s)) \qquad : s/(np\backslash s)$$

$$\lambda Q.(Q\ \lambda x.((\mathbf{see}\ \mathbf{m})\ x))\ (\lambda P.\neg\exists\lambda y.(P\ y)) \rightsquigarrow^*_\beta \neg\exists\lambda y.((\mathbf{see}\ \mathbf{m})\ y)$$

This decomposition of the ω-operator leads to a correct derivation. The wh-phrase *'who'* is an argument wh-phrase that expects a gq as an answer, even though the hypothesis gap in the question is of type np. A generalized quantifier, however, is a *lifted np*. Consequently, answers to a wh-question starting with who can be either generalized quantifiers, such as *'nobody'* or *'everybody'*, or noun phrases, such as *'John'* or *'Mary'*. Noun phrases, however, first have to be lifted to a higher type to be a general quantifier. Only then the semantic term of a noun phrase such as *'John'* that is lifted to $(\lambda P.\ (P\ \mathbf{j}))$ can be beta-reduced to the correct term.

In general, argument wh-phrases, such as *'what'* and *'who'*, expect a referential or quantified noun phrase as an answer that either refer to a single entity (*'John'*) or a group of entities (*'John and Mary'*, *'everybody'*). Wh-phrases that are *which*-determiners, though, only expect a referential noun phrase, in the sense that they, in combination with a noun, expect a unique entity as answer. They do not accept answers that refer to groups of entities. Generalized quantifiers, therefore, do not count as answers to *which*-determiners, as follows from the wrongly answered question *"Which man saw Mary? Everyone"*. *Which*-determiners require a syntactical type and semantic decomposition for the ω-operator to ensure that only unique entities count as answers. Vermaat decomposes the ω for *which*-determiners as follows:

$$\textbf{which} \vdash \lambda V.\lambda P.\lambda x.(x = \iota y.((V\ y)\ \wedge\ (P\ y))) : \mathrm{WH}(np, A, s/_?np)/n \quad \text{where } A \in \{s, q\}$$

As we can see, the syntactical side of *which*-determiners requests a noun on the right-hand side, before it can be used as a wh-phrase. For instance, the wh-phrase *"Who"* in *"Who saw Mary?"* can be substituted by *"Which man"* as they are both wh-phrases. As is also apparent from the syntactical type, *which*-determiners request a noun phrase as answer. A generalized quantifier, such as *"Nobody"* is not a correct answer to the question *"Which man saw Mary?"*. A noun phrase, such as *"John"*, that is lifted to function as a generalized quantifier, can therefore also not be accepted as an answer. The non-lifted version of noun phrases should be accepted as answers, so the type assignment of **which** is adjusted accordingly. Furthermore, *which*-determiners combined with a noun on the right-hand side, are able to fill the gap hypothesis of type np in either question bodies, q, or sentential bodies, s.

Semantically, a *which*-determiner denotes a function that takes two predicates, such as **man** and **(see m)**, and an entity, such as **j** for *'John'*, as arguments and returns a proposition. Note that, instead of a normal λ, an ι is used to construct the proposition. The following sample derivation shows how a *which*-determiner is used:

$$\dfrac{\dfrac{\dfrac{\textbf{which} \vdash \lambda V.\lambda P.\lambda x.(x = \iota y.(V\ y)\ \wedge\ (P\ y)) \quad : \mathrm{WH}(np, s, s/_?np)/n \qquad \textbf{man} \vdash \textbf{man} : n}{\textbf{Which} \circ \textbf{man} \vdash \lambda P.\lambda x.(x = \iota y.((\textbf{man}\ y)\ \wedge\ (P\ y))) : \mathrm{WH}(np, s, s/_?np)}\ [/E] \qquad \dfrac{\dfrac{\textbf{saw} \vdash \textbf{see} \quad : (np\backslash s)/np}{\textbf{saw} \circ \textbf{Mary} \vdash \textbf{see m} : np\backslash s}\ [/E] \qquad \dfrac{\textbf{Mary} \vdash \textbf{m} \quad : np}{np \vdash \textbf{x} : np}\ [\backslash E]}{(\textbf{Which} \circ \textbf{man}) \circ (\textbf{saw} \circ \textbf{Mary}) \vdash \lambda x.(x = \iota y.((\textbf{man}\ y)\ \wedge\ ((\textbf{see m})\ y)) : s/_?np}\ [\mathrm{WH}^l_{ex}] \qquad \dfrac{\textbf{John}}{\textbf{j} : np}}{((\textbf{Which} \circ \textbf{man}) \circ (\textbf{saw} \circ \textbf{Mary})) \circ_? \textbf{John} \vdash \textbf{j} = \iota y.((\textbf{man}\ y)\ \wedge\ ((\textbf{see m})\ y) : s}\ [/E]$$

In this derivation, we observe that the wh-phrase *"which"* combines with a noun, before it can substitute the hypothesis gap in the body of type s. After combining with the answer *"John"*, the semantic interpretation constitutes a proposition stating that John is the only man that saw Mary.

3.4 Conclusion

With these syntactical and semantic definitions of wh-phrases in the typelogical grammar of Chapter 2, we have now laid the foundations to start looking at incrementality within the interpretation of wh-phrases. The wh-type schemata defined by Vermaat can be used to model wh-phrases in the typelogical grammar. For this thesis, that means that the next problem is how to guarantee incrementality for the typelogical grammar. In Chapter 4, we'll investigate incrementality in general and in Chapter 5 we discuss how to apply it to the typelogical grammar. After we have found a way to guarantee incrementality for the typelogical grammar, we can move on and model incrementality for wh-questions using the wh-type schemata.

In this chapter, however, we have presented numerous different kinds of wh-questions, i.e.

(in)direct local wh-questions, non-local wh-questions, wh-questions with island constraints and multiple wh-questions. Incremental interpretation might be desired for all of these questions, but this thesis will focus on direct local wh-questions mostly. The reason for this is that the incremental algorithm that we will propose in Chapter 5 should be general enough to interpret almost any kind of wh-question. Moreover, what is not subject in this thesis is the types of wh-phrases that exist. In this chapter we have seen wh-phrases such as *'who(m)'*, *'what'* and *'which'* in the examples that have been provided. There are, however, more wh-phrases that should be taken into account. For example, the wh-phrases *'where'*, *'why'* and *'how'* might also be an interesting subject of research, as they require a different kind of answer than the referential wh-phrases *'who(m)'*, *'what'* and *'which'*. Before those wh-phrases can be incorporated in the incremental algorithm we will present, more research is necessary on the syntactic and semantic values of those wh-phrases.

What's more is that the syntactic and semantic value of the in-situ wh-type schema proposed by Vermaat is a bit questionable. It makes sense to use a scope marker that moves to the position in the question where it takes scope over the clause that the wh-phrase is embedded in, but the semantic term corresponding to the in-situ wh-phrase makes use of a designated variable $\mathtt{X}$ that occurs in both elements of the pair $\langle \lambda y.\ (\omega\ \lambda \mathtt{X}.\ y),\ \mathtt{X} \rangle$. The idea is when this variable changes in either element of the pair, the other changes as well. This concept of the designated variable is highly unusual. Moreover, the syntactic value of an in-situ wh-phrase, $\Diamond\Box(C/B) \bullet A$, makes use of the product operator. This is also quite an unusual type assignment. The incremental algorithm we will propose, will not be able to accommodate product operators, and therefore multiple wh-questions will also be left out of scope, as they are the only kind of wh-questions that contain in-situ wh-phrases.

Also, it is explained in this chapter how answers to wh-questions might be accounted for with the wh-type schemata. In the remainder of this thesis, however, answers to wh-questions will not be taken into account as they are not relevant to the subject of incremental interpretation. An answer will always be combined with the question to form a proposition after the question has been fully interpreted, in both incremental and non-incremental interpretation of wh-questions, and therefore we can leave accounting for answers out of scope in this thesis. What we do take into consideration in this chapter is the decomposition of the ω-operator. That is a useful piece of knowledge in the semantic term of a question, as we can examine it and know how and when the answer would be incorporated in the semantic term if it was present.

The most important aspect that should be taken away from this chapter, is the fact that wh-phrases are defined for the typelogical grammar of Chapter 2 and we can use their derivation rules to account for their behaviour in natural language. The next problem to tackle is how incrementality could be guaranteed in the typelogical grammar that is able to account for wh-phrases.

Chapter 4

Incrementality

4.1 Introduction

Incremental interpretation is the concept of interpreting sentences from left-to-right, word-by-word. Evidence has been found in psycholinguistics that humans interpret sentences incrementally (Marslen-Wilson [1973], Just and Carpenter [1980], Altmann and Steedman [1988], Kamide and Haywood [2003], Kuperberg et al. [2003]), meaning that humans have the ability to respond to a sentence even before the sentence has been uttered completely. An example of this we have seen before is the following dialogue:

Person A: *I think I have a pretty extensive, uh...*
Person B: *Vocabulary?*
Person A: *Yes!*

We observe that B was able to complement A's sentence based on the partial phrase preceding the last word.

In this thesis, *interpretation* of a sentence is assumed to consist of only the semantic composition of the individual words of the sentence. *Parsing*, on the other hand, is the syntactic analysis of the sentence. These two go hand in hand as semantic interpretation of a sentence can happen simultaneously with parsing that sentence, as we have seen in the derivation rules of typelogical grammars. Moreover, the concept of interpretation presupposes parsing, as a semantic composition does not exist without a syntactic analysis. For clarity purposes, we'll use this distinction. Whenever the term parsing is used, we refer to the process of connecting phrases to each other syntactically. Whenever the term interpreting is used, we are referring to the process of generating a semantics for the partial sentence thus far.

Besides the distinction between parsing and interpretation, the concept of *constituency* has to be discussed as well. The concept of constituency is the idea that sentences are built up from blocks, that in their turn can be built up from even smaller blocks. Usually, these blocks are binary branching, i.e. a constituent is built from two smaller constituents. This idea of constituency returns in most linguistic formalisms, such as Context-Free Grammars, Combinatory Categorial Grammars and typelogical grammars. From the derivation rules for the typelogical grammar

introduced in Chapter 2, for instance, follows that a transitive verb combines with the object of a sentence before it combines with the subject. Let's assume the sentence *"John loves Mary"*, where *'John'* and *'Mary'* are of type np and *'loves'* is of type $(np\backslash s)/np$, is to be interpreted. The verb *'loves'* combines with the object *'Mary'* first, since the type of *'loves'* expects a phrase of type np on its right-hand side. Combining these phrases results in the partial sentence *"loves Mary"*. This is called a constituent. Combining the result with *'John'* now results in the full sentence *"John loves Mary"* of type s. The concept of constituency therefore often blocks a linguistic formalism from guaranteeing incrementality. A full sentence is also a constituent, but we'll mostly use the concept of constituency in the context of partial sentences, where there are still words left to interpret. Note that given that interpretation also takes the semantic values of each phrase into account, every constituent will have its own semantic term corresponding to it. In this thesis, one of the challenges to overcome is how to reach incremental interpretation without changing the types in the lexicon to be incremental. In our running example, this means that the phrase *"John loves"* is the desirable phrase to be interpreted, even though *'loves'* still has a type that combines with the object first. We want to be able to interpret such non-constituents.

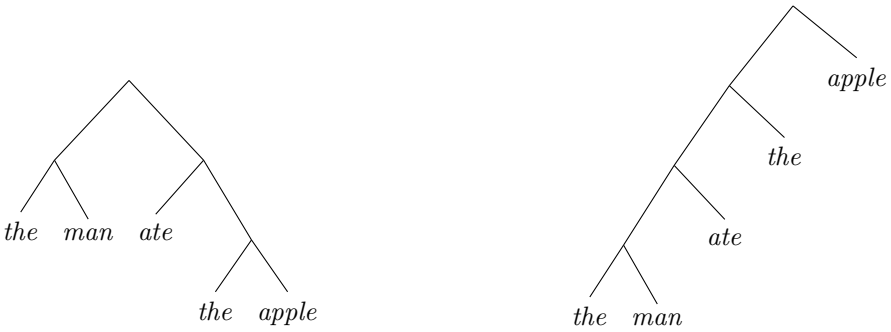
Incremental interpretation comes in two flavours, namely strong and weak incremental interpretation. Strong incremental interpretation maintains a fully connected structure of the input sentence during interpretation, i.e. every initial part of the sentence is interpreted as a constituent. For example, the first three words of the sentence *"John thought that Bill is the author of this paper."* are *"John thought that"* and is, in the strongly incremental paradigm, the desired constituent. It is assumed by Lombardo and Sturt [2002] that human interpreting is strongly incremental. They do not, however refer to any theories that exemplify this.

Opposite from strong incremental interpretation is weak incremental interpretation, which does not interpret every word in the input sentence as it is inserted. Weak incremental interpretation allows waiting for one or more input words before connecting phrases to each other. For example, when a determiner such as *'the'*, *'every'* or *'which'* is interpreted, weak incremental interpretation is allowed to wait for and connect the determiner to the noun that follows the determiner to build a determiner phrase, before connecting the determiner phrase to the partial sentence preceding it. The phrase *"John hates every"* is an instance of a partial sentence that has a corresponding partial structure and semantics when interpreting it strongly incrementally as every input word is connected to the preceding phrase. This phrase has no corresponding structure or semantics when interpreting it weakly incrementally, for weak incremental interpretation requires an extra noun phrase after *'every'*. Say, *"John hates"* is parsed and is of type s/np, while *'every'* is of type np/n and requires a noun on its right-hand side. Weak incremental interpretation does not allow *'every'* to be parsed at this point, since *"John hates"* requires a noun phrase on its right-hand side, and *'every'* is not a noun phrase. It is an incomplete noun phrase with respect to a noun on the right. Weak incremental interpretation now allows *'every'* to complete itself by getting a noun on the right and form a noun phrase. Weak incremental interpretation may be regarded as interpretation with a look-ahead function that waits until enough input words are processed before a structure is generated.

Strong incremental interpretation has two advantages. First, it allows for the interpretation of partial sentences and therefore an incomplete utterance can be understood as well, no matter what that sentence may be. And secondly, according to Lombardo and Sturt [1997] strong incremental interpretation is memory-efficient as no memory is needed for not yet connected phrases, i.e. memory is not needed in strong incremental interpretation, since every incoming input word

is directly connected to the preceding phrase. Naturally, the advantages of strong incremental interpretation do not hold for weak incremental interpretation. Weak incremental interpretation, on the other hand, does have the advantage that incomplete constituents, such as *"John likes every"* can't be interpreted, as the word *'every'* expects a predicate, such as *'man'* or *'woman'* first.

In this thesis, we will understand incremental interpretation as the strong variant of incrementality. To illustrate this we draw a tree of a non-incremental derivation we have seen before and a tree of the same sentence but derived incrementally. The example sentence is *"The man ate the apple"*. The trees will look as follows, where the non-incremental tree is on the left and the incremental tree on the right:



As opposed to the non-incremental tree, in the incremental tree the phrase *"the man"* is combined first after which it combines with the remainder of the sentence word-by-word from left-to-right. The non-incremental tree combines the constituent *"the man"* with the entire constituent of *"ate the apple"*, which is not incremental, as *"ate the apple"* has to be derived before combining. Most importantly, what we expect from incremental interpretation is an unknown future input. At every point where the tree branches in two, we expect a semantic term corresponding to the until then interpreted input. That semantic term takes up one argument, namely the next word. Providing every next word as input for the interpreted semantic term will be eventually yield the semantic term of the entire sentence. This is what we will understand as incremental interpretation in this thesis.

Research into incremental interpretation is usually focused upon parsing and a variety of algorithms for parsing, whether or not incremental, have been developed. For example, Earley's algorithm [Earley, 1970], the CYK algorithm [Younger, 1967] and LL and LR parsers (Stearns and Lewis [1969] and Knuth [1965]), are all parsing algorithms for natural language. In this chapter, we'll treat algorithms and formalisms as guaranteeing incrementality when they are deterministic and the result of interpretation is always an incremental interpretation, i.e. an interpretation from left-to-right, word-by-word.

All algorithms that can be used to parse sentences rely on grammar formalisms, such as Context-Free Grammars (CFG), Combinatory Categorial Grammars (CCG) or Categorial Grammars (CG). Most of the well-known parsing literature researches parsing on the basis of CFG's. But when incrementality comes into play, the literature focuses on CCG's mostly, as seen in Demberg [2012], Zhang and Clark [2011] and Ambati et al. [2016]. CCG's are flexible when it comes to

their derivation rules. A new rule may always be added when it is proven to be correct. This entails that the set of possible derivations for a sentence may be infinite. CCG's also have the property of binary branching. That property is important as there will be no need to ever combine more than two phrases. When combining three or more phrases, it has to be explicit which phrase is the function and which phrases the arguments, but in CCG's and CG's this follows naturally from the types they are given. Of course, CFG's can be binary branching as well, but the fact that CCG's and CG's are inherently binary branching makes that they are formalisms that lend themselves very well for incremental interpretation. For example, when the first word of a sentence is taken to be a function that takes the following words as arguments one at a time and returns the syntactical type or semantics of the entire sentence. CG's on the other hand have a fixed set of derivation rules. This makes deriving sentences a lot less straightforward, but has the advantage that spurious ambiguity is avoided that way.

As mentioned before, this thesis uses typelogical grammars as the formalism to interpret sentences incrementally. Following from the fact that types are not inherently incremental, the derivation rules presented in Chapter 2 are not inherently incremental either. Therefore, in chapter 5 the system **M** due to Moortgat [1988] will be presented that can be used for modelling incremental interpretation. **M** calculates the *intermediate type* of two types. The intermediate type is a category that is the middle ground between two categories. For example, when the three categories np, $(np\backslash s)/np$ and np, corresponding to the three words *'John'*, *'saw'* and *'Bill'*, are inputted in that order, the first two words (*'John'* and *'saw'*) do not evaluate to a constituent according to the rules presented in Chapter 2. However, when the system **M** is used, the first two words evaluate to an intermediate type s/np. In this case, **M** pauses the application of the object to a transitive verb, and applies the subject to it's corresponding position in the verb to result in a verb that still takes an object to its right. Application of the intermediate type to the word *'Bill'* evaluates to the sentence type s. This thesis will combine a controlled variant of **M** and the wh-type schema to achieve incremental interpretation of wh-questions. The existence of **M** might be useful to keep in mind while reading this chapter.

This chapter is divided into two parts and will mostly be a descriptive chapter, exploring the concept of incrementality and how it follows (or not) from different formalisms. Firstly, incrementality in different formalisms will be discussed. Then, two other approaches that might be useful to have knowledge of in the context of incremental interpretation are discussed.

4.2 Incrementality in different formalisms

Research into incremental interpretation has focused itself mostly on parsing instead of interpretation and in this section a few of the frameworks and algorithms that parse incrementally will be discussed. First, we will discuss incremental parsing using Context-Free Grammars (CFG). And finally, incremental parsing using the Combinatory Categorial Grammar-framework (CCG) is discussed together with a brief comparison with typelogical grammars. After that, we still have some approaches that could be useful in reasoning about incremental interpretation which we will discuss as well. These approaches include an 'incremental constituent' approach and a continuation-based approach.

4.2.1 Frameworks

Context-Free Grammar

A CFG is a set of production rules that describes the set of sentences that can be generated with that partical CFG given a set of terminal symbols. The production rules are based upon the idea of constituents. That idea stems from the time of the Sanskrit philologist and grammarian Pāṇini, but the CFG was only developed in the 1950's by Noam Chomsky [Hopcroft, 1979]. The following sentence can be bracketed in a way the illustrates this idea:

Sentence:	Bill thinks Mary saw the bird that whistled
Bracketed:	(Bill (thinks (Mary (saw ((the bird) (that whistled))))))

Here, we observe that the entire sentence is a block that is built up from the parts *"Bill"* and *"thinks Mary saw the bird that whistled"*. The second part is in its turn again a block that is built up from 2 parts, etc. The phrase *"the bird that whistled"* is again built up from 2 parts, but these two parts constitute of multiple words, namely *"the bird"* and *"that whistled"*. It is important to note that this example is binary branching, but this is certainly not always the case.

Definition 18 (Context-Free Grammars) *A Context-Free Grammar is defined by the 4-tuple:*
$G = \langle V, \Sigma, R, S \rangle$*, where:*

1. V *is a finite set. Each element* $v \in V$ *is called a non-terminal. Non-terminals are used to specify what type of phrase is parsed or generated.*
2. Σ *is a finite set that is disjoint from* V*. Each element* $t \in \Sigma$ *is called a terminal and is used to signify a word in the language that is being modeled.*
3. R *is a finite relation between* V *and* $(V \cup \Sigma)^*$*, where* * *is the Kleene-star operation. Each element of* R *is a production rule, generally written as* $X \to Y$*, where* $X \in V$ *and* $Y \in (V \cup \Sigma)^*$*.*
4. $S \in V$ *is a non-terminal that is used as start symbol. A start symbol represents the entire phrase that is parsed or generated.*

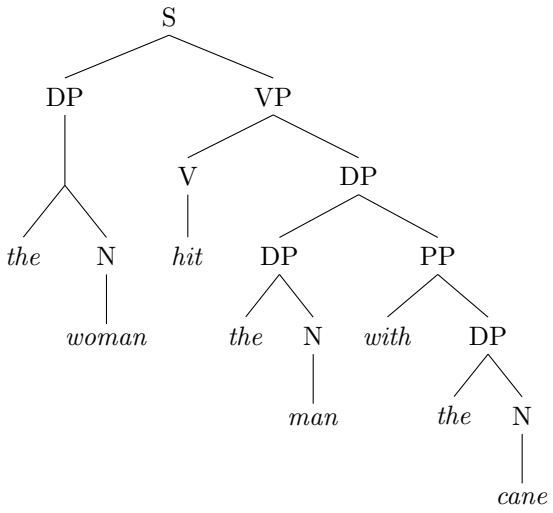
A CFG parses a sentence by starting with the start symbol and recursively rewriting non-terminals with help of the production rules until no non-terminals are left. When a non-terminal is rewritten the non-terminal is replaced by the right-hand side of the production rule. We illustrate this principle when the CFG is defined by the tuple $\langle \{S, DP, N, NP, PP, V, VP\}$, $\{the, with, man, woman, cane, hit\}, R, S \rangle$, where R is the following set:

1 S → DP VP
2 VP → V DP
3 DP → *the* N
4 DP → DP PP
5 PP → *with* DP
6 N → *woman* | *man* | *cane*
7 V → *hit*

We find the following sequence of rewriting for for the sentence *"The woman hit the man with the cane."*:

Rewrite rule	Stack
Start	S
1	DP VP
3	*the* N VP
6	*the woman* VP
2	*the woman* V DP
7	*the woman hit* DP
4	*the woman hit* DP PP
3	*the woman hit the* N PP
6	*the woman hit the man* PP
5	*the woman hit the man with* DP
3	*the woman hit the man with the* N
6	*the woman hit the man with the cane*

In the end, no non-terminals are present in the stack, so parsing is complete. The sentence is correctly generated by the CFG. Note that we have expanded the non-terminals from left-to-right, which is not a necessary condition. We could easily have expanded the non-terminals in a different order and end up with the same result. We can draw the structure of this sentence as the following tree:



Incrementality in interpretation in the context of CFG's is not similar to incrementality in interpretation in the context of other formalisms. The incrementality found in the previous example is coincidental and purely stems from the order in which non-terminals are expanded and the definition of the production rules. Had we chosen a different order to expand non-terminals, we would have found a different order in which words had been interpreted. There is no function-argument structure as is present in CCG's and CG's But that is not the only problem that arises when incrementality is required from CFG's. To parse incrementally, a CFG has to have an idea of the structure that will follow the current input word. The most common example of this is

left-recursion. Imagine adjectives before a noun. There is no limit to the amount of adjectives that can occur before a noun, and therefore, when the first adjective has been parsed, there's no way of knowing how many will follow. A CFG that has multiple choices when expanding a non-terminal, i.e. a non-deterministic CFG, has to predict what the following structure will be, and since there could be an infinite amount of adjectives, this is always a viable option to predict. This problem is called Infinite Local Ambiguity. Lombardo and Sturt [1997] give two examples of parsers in which this problem is overcome with use of the concept of Minimal Recursive Structure.

Lombardo and Sturt [1997] define 5 steps their parsers go through in the process of parsing, namely initialization, prediction, scanning, completion and termination. In short, initialization starts with the start symbol of the CFG and places a 'dot' in front of the first symbol on the right-hand side of the start symbol production rule. If that first symbol is a non-terminal, prediction takes place, in which a possible structure is predicted for that non-terminal and the dot is placed in front of the first symbol on the right-hand side of the predicted production rule. If a terminal is encountered after the dot, the dot proceeds to scan whether there exists a terminal production rule in the CFG that corresponds to that terminal. If there is such a rule, the dot is placed behind the scanned terminal. If there isn't such a rule, backtracking is activated and a new prediction must take place. Completion is the step that finishes scanning a predicted production rule. In this case the dot is placed behind the brackets that followed from the prediction rule. Termination is the final step which accepts when the dot is positioned after the brackets of a $[_S ...]$ form. Otherwise it rejects. Intuitively, in this context, a Minimal Recursive Structure is a marker when left-recursion is encountered. The structure that has been parsed at that point is minimal and parsing will not continue, unless there is reason to do so. We will illustrate these parsers in action with the following example.

Example Say the sentence *"John loves Mary"* is to be parsed using the following CFG:

$$
\begin{array}{ll}
1 & \text{S} \rightarrow \text{NP VP} \\
2 & \text{VP} \rightarrow \text{v NP} \\
3 & \text{NP} \rightarrow \text{pn}
\end{array}
$$

Note that v and pn are any words in the set of verbs and proper nouns, respectively. We now use the 5 defined steps in the following order to parse the example sentence:

1. INITIALIZATION
$[_S\ [_{NP}$ .pn] VP]
2. SCANNING
$[_S\ [_{NP}$ John.] VP]
3. COMPLETION
$[_S\ [_{NP}$ John] .VP]
4. PREDICTION
$[_S\ [_{NP}$ John] $[_{VP}$ .v NP]]
5. SCANNING
$[_S\ [_{NP}$ John] $[_{VP}$ loves .NP]]
6. PREDICTION
$[_S\ [_{NP}$ John] $[_{VP}$ loves $[_{NP}$ .pn]]]
7. SCANNING
$[_S\ [_{NP}$ John] $[_{VP}$ loves $[_{NP}$ Mary.]]]

8. COMPLETION (3x)
$[_S\ [_{NP}$ John] $[_{VP}$ loves $[_{NP}$ Mary]]].

As we see, the 5-step parsing method of Lombardo and Sturt is a rather easy procedure and, moreover, incremental, as it substitutes non-terminals from left to right. It is, however, subject to the dangers of prediction. In a left-recursive CFG, these parsers may end up in an infinite loop of prediction.

Combinatory Categorial Grammar

The formalism of Combinatory Categorial Grammars (CCG) has been developed by Steedman [2000] in the early 2000's. It relies on combinatory logic, hence the name. Combinatory logic has the same expressive power as lambda-calculus and is therefore useful in modeling natural language. CCG is very similar to typelogical grammars as it is a lexicalized grammar, in the sense that every word has its own category built up out of basic types. Adding to that, it also consists of a small set of *derivation rules* that are used to derive axioms. The difference between the two formalisms is that CCG is not limited in the rule set. Derivation rules can be added to account for certain phenomena in syntactical structures, meaning that there are potentially infinitely many derivations for a certain sentence. This leads to spurious ambiguity. Since typelogical grammars do have a fixed rule set, spurious ambiguity will not arise so easily.

A category in CCG still signifies a class of words that behave similarly in the particular language. The notation of both the categories and derivations in CCG is different, however, from typelogical grammars. In CCG, a category is again a basic type, such as s, np, n, or even pp, or a complex category. The category-building operations are / and \, which again signify incompletion to the right and left, respectively. A category Y that is incomplete to its right with regard to a category X is again written as Y/X. A category Y that is incomplete to its left with regard to a category X, however, is written as $Y\backslash X$. This is a small notational difference between CCG and typelogical grammar, but keep it in mind while reading this section. For example, a transitive verb would now be written as $(s\backslash np)/np$ for CCG's.

Considering the derivation rules, they do not have a sequent style presentation like in the typelogical formalism. They are based on function application and composition and can differ per language. For each rule we have two categories that are combined under the application of a derivation rule. Once again, we observe a similarity with typelogical grammars. We have the following rules for CCG's for subject-verb-object ordered languages, such as English:

Forward Application:	X/Y	Y	$\Rightarrow_{>}$	X
Backward Application:	Y	$X\backslash Y$	$\Rightarrow_{<}$	X
Forward Composition:	X/Y	Y/Z	$\Rightarrow_{>B}$	X/Z
Backward Composition:	$Y\backslash Z$	$X\backslash Y$	$\Rightarrow_{<B}$	$X\backslash Z$
Forward Generalized Composition:	X/Y	$(Y/Z)/\$_1$	$\Rightarrow_{>B^n}$	$(X/Z)/\$_1$
Backward Crossed Composition:	$Y\backslash Z$	$X\backslash Y$	$\Rightarrow_{<B_x}$	X/Z
Forward Type-raising:	X		$\Rightarrow_{T}$	$T/(T\backslash X)$
Coordination:	X conj	X	$\Rightarrow_{\phi}$	X

The derivation rules are used in the same manner they are used in typelogical grammars. By

applying functions to arguments, the sentence type should eventually be derived. A difference between the two formalisms, is that CCG does not derive until axioms are found and does not know rules that resemble the introduction rules in typelogical grammars. We provide an example derivation of the sentence *"Everybody likes the man."*:

$$\dfrac{\dfrac{\text{Everybody}}{S/(S\backslash NP)}\quad\dfrac{\dfrac{\text{likes}}{(S\backslash NP)/NP}\quad\dfrac{\dfrac{\text{the}}{NP/N}\quad\dfrac{\text{man}}{N}}{NP}>}{S\backslash NP}>}{S}<$$

In this example derivation we observe that incrementality is not achieved. We could, however, also have chosen to apply the Forward Composition derivation rule to *"Everybody"* and *"likes"*. This kind of non-determinism leads to spurious ambiguity in CCG. Spurious ambiguity is problematic for incrementality, since it would mean that an incrementally interpreted sentence has a different semantic term corresponding to it, just because it is interpreted incrementally. It is, however, desirable to get the semantic term equal to the semantic term that is the result of not incrementally interpreting the sentence (leaving sentences that have multiple derivations and therefore also multiple semantic terms, such as *"Everybody likes someone"* out of consideration). Another disadvantage of CCG is, according to Demberg [2012], the fact that strong incrementality is not possible within CCG parsing, unless the internal structure of categories is adjusted and the following rule is added to the rule set:

$$\textbf{Geach rule: } Y/Z \Rightarrow_{\mathbf{B}} (Y/G)/(Z/G)$$

Zhang and Clark [2011] did however find an incremental CCG parser that parses sentences weakly incremental. It is based on the idea of Shift-Reduce Parsing, which uses a stack to shift nodes onto and reduces the stack whenever two nodes can be reduced to one node. Their Shift-Reduce CCG Parser has 4 actions, namely: SHIFT, COMBINE, UNARY and FINISH. Next to these actions they keep track of a stack. When a new word is given as input, the SHIFT-X action is taken. This action pushes the word onto the stack and assigns the category X to it. When two nodes are able to combine by any CCG derivation rule, a reduce-step can be taken, which corresponds to the COMBINE-X action. It pops two nodes off the stack and pushes the result node onto the stack. The resulting node should be of the category X. Whenever a category is not adequate for any of the derivation rules, the UNARY-X action is always possible for use. It corresponds to the Forward type-raising rule, and raises a category of, for example, NP to the complex category $S/(S\backslash NP)$. Once again, the result of UNARY-X should be of the category X. Finally, the FINISH action terminates parsing. It can happen at any time, even when there are multiple nodes left on the stack, in which case it produces multiple partial derivation trees.

Example To illustrate the procedure of Shift-Reduce parsing, we'll parse the example sentence *"The girl wanted a balloon."*. We show the stack of shifted nodes, queue of input nodes to come and the action taken at every step[1]:

[1]We use the symbol ϵ to represent the empty stack or queue. We have also noted which CCG derivation rule has been used for clarity. $FA_{>}$ represents Forward Application, whereas $BA_{<}$ represents Backward Application

	Stack	Queue	Action
1.	ϵ	NP/N, N, $(S\backslash NP)/NP$, NP/N, N	Initialization
2.	NP/N	N, $(S\backslash NP)/NP$, NP/N, N	SHIFT
3.	NP/N, N	$(S\backslash NP)/NP$, NP/N, N	SHIFT
4.	NP	$(S\backslash NP)/NP$, NP/N, N	COMBINE, $FA_{>}$
5.	NP, $(S\backslash NP)/NP$	NP/N, N	SHIFT
6.	NP, $(S\backslash NP)/NP$, NP/N	N	SHIFT
7.	NP, $(S\backslash NP)/NP$, NP/N, N	ϵ	SHIFT
8.	NP, $(S\backslash NP)/NP$, NP	ϵ	COMBINE, $FA_{>}$
9.	NP, $S\backslash NP$	ϵ	COMBINE, $FA_{>}$
10.	S	ϵ	COMBINE, $BA_{<}$
11.	S	ϵ	FINISH

We observe that all categories are shifted to the stack at one point, and consequently combined with the already present categories on the stack. The result is a category S (on the stack) for the entire sentence (empty queue).

The Shift-Reduce CCG Parser of Zhang and Clark does not guarantee incrementality, since it is not obligated to use the COMBINE action whenever the stack contains exactly two nodes. It might just as well wait for more nodes, before combining the nodes, as in the example above and therefore does not guarantee incrementality. Also, the Shift-Reduce Parser has difficulty with parsing prepositional phrases at the end of the sentence. For instance, when parsing the sentence *"John likes mangoes from India"*, the Shift-Reduce Parser is able to stop parsing after *"John likes mangoes"*. At that point the prepositional phrase is still left in the queue. The parser will generate a partial derivation for the prepositional phrase as well, after which it will return two partial derivations. It is desirable, however, to return one derivation of the entire sentence. Ambati et al. [2015] noticed this and constructed a new parser that is similar to the CCG Parser of Zhang and Clark, but has two extra actions for *revealing*, LEFT REVEALING and RIGHT REVEALING. Intuitively, these actions act before the COMBINE action and wait for more information to be connected to the thus far parsed phrase. They abstract over the necessary phrase for the current category, so the expected category can be combined with the current category, before combining the current category with the preceding phrase. The difference between these two revealing actions is the way they are applied. For example, LEFT REVEALING is used for VP coordination, and RIGHT REVEALING is used for NP coordination and PP attachment. They achieve weak incrementality with this parser.

The framework chosen for this thesis is typelogical grammars, however. It has the advantage of having a strict rule set, and spurious ambiguity is therefore avoided. Every distinct derivation has a different semantic term corresponding to it. Moreover, the wh-type schemata due to Vermaat have been defined for typelogical grammars, while CCG's do not have such schemata. The schemata for typelogical grammars can therefore also be used in this thesis. But the typelogical grammar as we have presented in Chapter 2 does not guarantee incrementality. To make sure incrementality is guaranteed we add an algorithm that 'derives' an intermediate type that combines two types even though they perhaps couldn't combine by function application. But first, we turn to two other approaches that might help in reasoning about the problem to be solved.

4.2.2 Other approaches

Incremental constituents

As stated before, incrementality does not follow from the derivation rules of typelogical grammar formalisms. The concept of constituency, that is regarded in the syntactic type definitions of different words, prevents the typelogical grammar as we introduced in Chapter 2 from being incremental. In this subsection the following sentence and bracketing of constituents is used as a running example:

(Bill (likes ((a girl) (that (is blonde)))))

The phrase *"likes a girl that is blonde"* is a constituent. The verb *'likes'* has indeed connected with its object *'a girl that is blonde'* first. This is reflected in the type definition of transitive verbs, being $(np\backslash s)/np$. The np on the right-hand side of the verb, the object, is expected to combine first with the verb. This throws a spanner in the works for incrementality.

However, if we disregard this type definition, we are able to define our own 'incremental types'. A transitive verb would, for example, not be of type $(np\backslash s)/np$ anymore, but of type $np\backslash(s/np)$. In this incremental type we find that the verb should combine with the np on its left-hand side first, which is the subject. We find the following incremental derivation for the sentence *"John saw Bill"* when using the incremental type for the verb:

$$\dfrac{\dfrac{\overline{\mathsf{John} \vdash np} \quad \overline{\mathsf{saw} \vdash np\backslash(s/np)}}{\mathsf{John} \circ \mathsf{saw} \vdash s/np}\,[\backslash E] \quad \overline{\mathsf{Bill} \vdash np}}{\mathsf{John} \circ \mathsf{saw} \circ \mathsf{Bill} \vdash s}\,[/E]$$

One would think that incrementality has been achieved with this approach. This is not exactly the case, however. For instance, in the running example sentence *"Bill likes a girl that is blonde."* we found that the noun phrase *'a girl'* is a constituent. The word *'a'* is of type np/n and *'girl'* of type n. The first two words of this sentence can be incrementally interpreted to result in the type s/np, similar to what we have seen in the above derivation. But since we now have a determiner of type np/n instead of a noun phrase of type np, the word *'a'* cannot be given as an argument to the result type of the first two words. The derivation has to wait until the word *'girl'* is supplied as an argument to *'a'* before further application can take place. We illustrate this in the following derivation:

$$\dfrac{\dfrac{\overline{\mathsf{Bill} \vdash np} \quad \overline{\mathsf{likes} \vdash np\backslash(s/np)}}{\mathsf{Bill} \circ \mathsf{likes} \vdash s/np}\,[\backslash E] \quad \dfrac{\overline{\mathsf{a} \vdash np/n} \quad \overline{\mathsf{girl} \vdash n}}{\mathsf{a} \circ \mathsf{girl} \vdash np}\,[/E]}{\mathsf{Bill} \circ \mathsf{likes} \circ \mathsf{a} \circ \mathsf{girl} \vdash s}\,[/E]$$

In this derivation, the constituent *'a girl'* has to be derived before it can be supplied as an argument to the constituent *'Bill likes'*. This observation makes this approach weakly incremental at best.

Continuations

Another concept that deserves a mention in the context of this thesis is the concept of continuations. A continuation is originally used in computer science and is an abstract representation of the computational process. It has been 'reinvented' several times throughout the history of computer science, due to a little bit of poor communication and the many different fields in which continuations were eventually useful [Reynolds, 1993]. The first description of continuations was given by Adriaan van Wijngaarden in September 1964 at the IFIP Working Conference on Formal Language Description Languages. He formulated a preprocessor to translate the programming language Algol 60 into a sub-language. The final step of preprocessing is what we now call the transformation to Continuation-Passing Style, which is the style of writing functions with continuations.

To get into what continuations actually are, Lebedeva [2012] defined continuations as the remainder of the computation after evaluating an expression c when the evaluation of a certain program was in a state s. Basically, a continuation can be seen as the collection of future computations. However, Lebedeva focuses mainly on artificial languages it seems, and in this thesis we are interested in natural languages.

Lebedeva based her work on earlier work by De Groote [2006]. He provided a very general framework that can be used to find meaning in discourse, i.e. to deduce the semantics of a sentence by looking at its *context*. The continuation in this framework is described as the *right context* of a certain expression, meaning that the semantic value of a sentence depends on what's to come. De Groote also provided types for the context and described how to work with it when interpreting discourse.

Barker and Shan [2014] defined continuations as a portion of text surrounding a natural language expression. For example, the continuation in the following sentence contains the boldfaced words in the bracketed clause:

*John said [**Mary called** everyone **yesterday**] with relief.*

Barker and Shan abstract over the *scope-taker* 'everyone', as it is this lexical element they want to know the continuation of, and are left with the continuation that has a gap in the place where the scope-taker used to be. To illustrate, the corresponding semantic value to this continuation is $\lambda x.\ \textsc{yesterday}(\textsc{called}\ x)\ \mathbf{m}$, where the gap of the scope-taker is provided by the lambda. This style of continuation looks like it can be very useful when interpreting wh-questions. When we abstract over the wh-phrase in a wh-question, we are left with, how Vermaat calls it, the body of the question. That body might also be seen as a continuation of the wh-phrase, where the wh-phrase creates a gap in the semantics of the wh-question.

Looking at it from an incremental point of view, the concept of continuations is a useful concept to have knowledge of. In the ideal world, we would want to abstract over what input word is next and take an input word as an argument to the semantic function that we're building for a question. Moreover, we can leave the answer to a question out of scope in this thesis in this case, since it can be seen as just another continuation of the question that will form a proposition when given. We will see that the system **M** that is discussed in the following section, ends up with semantic terms similar to the semantic terms that would follow from the continuation-based

approach.

In this chapter we have mostly described formalisms and algorithms that may or may not guarantee incrementality. Knowledge of these formalisms and algorithms might provide an insight into incrementality that helps us find the correct semantic terms of partial sentences. In Chapter 5, we will present the system of **M** that combines two typelogical types. That intermediate type must have a corresponding semantic term, that **M** also derives for us. We will find that that when the intermediate type is not the type of the entire sentence, it will require an argument (bound by a lambda) that resembles a continuation.

Chapter 5

An incremental algorithm

5.1 Introduction

According to the incremental constituents mechanism introduced above, the subject of a sentence was supplied as the argument to the verb before the object was supplied as an argument. Although that approach did not lead to incrementality, the change in order of application is an interesting idea. What we are looking for is a way to combine types, and by that manipulate the order of application, even though they can not be combined according to the derivation rules of typelogical grammars. The system **M** due to Moortgat [1988] does exactly that. The system **M** takes as input two types and by recursively iterating through the rules of this system gives a type that is the middle ground between the two input types as output. That middle ground type is what we will call the *intermediate type*.

The system **M** is not part of the typelogical grammar rule set, but we can use it to find a type that together with the property of transitivity can be used in derivations. The property of transitivity is represented by the so-called *Cut-rule*. The Cut-rule is part of the set of derivation rules of the typelogical grammar, but is proven to be unnecessary as the other derivation rules are sufficient to derive a sentence without use of the Cut-rule. The Cut-rule is usually undesired as it introduces a variable that could have an infinite number of instantiations and therefore might heavily complicate the derivation. In this case, we will see that it is actually necessary to have the Cut-rule in our set of derivation rules to model incrementality. We will need to find common ground between two types, which will be the instantiation of the variable introduced by the Cut-rule.

Definition 19 (Cut) *If* Γ *derives a type* A *and* A *and* Δ *derive a type* B*,* Γ *and* Δ *derive* B*. Type* A *is what we call the intermediate type.*

$$\frac{\Gamma \vdash A \quad A, \Delta \vdash B}{\Gamma, \Delta \vdash B}\ [Cut]$$

In the remainder of this chapter we introduce the system **M** , provide a few examples of derivations using the cut-rule and present our version of the system **M** for incremental interpretation of wh-phrases.[1]

[1]Please note that the typelogical grammar and **M** are not the same. **M** is simply a procedure to find an

5.2 The system M

The system **M** provides three rules, *R1*, *M1* and *M2*, that by pattern matching searches for the intermediate type of two types. Each of these rules has a left and a right variant. If the intermediate type is not found by applying one of the rules, the system recursively iterates through its rules until an intermediate type has been found. We use different symbols to signify the difference between typelogical derivations ($\vdash$) and steps ($\Rightarrow$) of the system **M**.

Definition 20 (System M) *The system* **M** *provides three rules that each have their left and right variants, denoted by a subscript l or r, respectively.*

- *R1: Application*

  If the left (or right) type is of the form X/Y (or $Y\backslash X$) and the right (or left) type is of the form Y, apply the function type to the argument type to result in an intermediate type X.

$$R1_l: \ X/Y, Y \Rightarrow X$$
$$R1_r: \ Y, Y\backslash X \Rightarrow X$$

- *M1: Recursion on the range subtype*

  If the left (or right) type is of the form Z and the right (or left) type is of the form X/Y (or $Y\backslash X$), the resulting intermediate type is of the form W/Y (or $Y\backslash W$), where W is the intermediate type of Z and X (or X and Z).

$$M1_l: \ Y\backslash X, Z \Rightarrow Y\backslash W, \ \textit{where } X, Z \Rightarrow W$$
$$M1_r: \ Z, X/Y \Rightarrow W/Y, \ \textit{where } Z, X \Rightarrow W$$

- *M2: Recursion on the domain subtype*

  If the left (or right) type is of the form X/Y (or $Y\backslash X$) and the right (or left) type is of the form Z, the resulting intermediate type is of the form X/W (or $W\backslash X$), where Y is the intermediate type of Z and W (or W and Z).

$$M2_l: \ X/Y, Z \Rightarrow X/W, \ \textit{where } Z, W \Rightarrow Y$$
$$M2_r: \ Z, Y\backslash X \Rightarrow W\backslash X, \ \textit{where } W, Z \Rightarrow Y$$

These rules apply to two types at a time, meaning they can be chosen to apply to the first two types that are input, i.e. the first two words of a sentence. The resulting intermediate type can be chosen together with the third input word to result in an intermediate type of the first three words. This way of interpreting can be repeated until there are no more input words left. The final intermediate type is the sentence type.

To connect the system **M** with the typelogical derivations we need to clarify what the intermediate type exactly is in the typelogical counterpart. From the definition of the Cut-rule we can observe that an extra variable A is introduced in the premises. This variable is the intermediate

intermediate type that can be used in the Cut-rule in typelogical derivations.

type. If we choose Γ to be the first two types of a sentence, they will combine to an intermediate type that combines with the rest of the types of the sentence to form the sentence type. Basically, this makes the system **M** an efficient procedure to find the type of that variable A in the definition of the Cut-rule.

Example 1 Once again, let us take the sentence *"John saw Bill"*, where *'John'* and *'Bill'* are of type np and *'saw'* is the regular transitive verb type $(np\backslash s)/np$. The first two words are *'John'* and *'saw'* with types np and $(np\backslash s)/np$ in that order. Applying **M** on these two types results in the following calculations:

Step	Calculation	Rule
1	$np, (np\backslash s)/np \Rightarrow W/np$	$M1_r$
	$np, np\backslash s \Rightarrow W$	
	$W = s$	
2	$np, (np\backslash s)/np \Rightarrow s/np$	Conclusion

Step 1 is the application of $M1_r$ to the types we have at hand, namely np and $(np\backslash s)/np$. The second line in Step 1 is the condition of rule $M1_r$ and out of that condition we infer that $W = s$. Step 2 wraps up the calculation by substituting W by s, and we end up with the intermediate type s/np for the words *'John'* and *'saw'*. Now we have the intermediate type of the first two words, the third word, *'Bill'* of type np can be given as input. Applying **M** on the types s/np and np results in the following calculations:

Step	Calculation	Rule
1	$s/np, np \Rightarrow s$	$R1_l$

This calculation is just simple application by means of the $R1$ rule and therefore does not need to do further calculations. The resulting intermediate type is s for the phrase *"John saw Bill"*, which happens to be the sentence type, for the entire sentence. A typelogical derivation using the Cut-rule, would look like the following:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{John} \vdash np} \quad \dfrac{\dfrac{}{\mathsf{saw} \vdash (np\backslash s)/np} \quad \dfrac{}{np \vdash np}}{\mathsf{saw} \circ np \vdash np\backslash s}\,[\backslash E]}{\mathsf{John} \circ (\mathsf{saw} \circ np) \vdash s}\,[/E]}{(\mathsf{John} \circ \mathsf{saw}) \circ np \vdash s}\,[Ass]}{\mathsf{John} \circ \mathsf{saw} \vdash s/np}\,[/I] \quad \dfrac{\dfrac{}{s/np \vdash s/np} \quad \dfrac{}{\mathsf{Bill} \vdash np}}{s/np \circ \mathsf{Bill} \vdash s}\,[/E]}{\mathsf{John} \circ \mathsf{saw} \circ \mathsf{Bill} \vdash s}\,[Cut]$$

On the top left part of this derivation we observe the regular sentence derivation. What is actually happening is hypothetical reasoning, exactly like rule $M1$ is supposed to do. It pauses the application to the object for a moment by hypothesizing that there exists an object and is applied and continues to apply the subject first. After that, the hypothesis that the object was supplied is withdrawn and the real object (*'Bill'*) can be supplied to the intermediate type, which derives the sentence type s. Also note the use of associativity. All rules of system **M** make use of the associativity postulate behind the scenes.

Semantics of the system M

Naturally, the system **M** not only provides the syntactical point of view, but also the semantical point of view. We will present the semantic definitions that go together with the syntactical rules in a more schematic way than we have done before. The procedure of **M** and finding the semantic term corresponding to the intermediate type is, nevertheless, equal to finding only the syntactic intermediate type, by way of substituting the variables with known types and terms. We use Functor to denote the function that will take an argument, Arg to denote arguments and SemX and VarX to denote the semantic term and semantic variable corresponding to syntactic variable X, respectively. The symbol λ denotes abstraction as usual and Functor Arg denotes function application, where Functor is the functor and Arg the argument.

Definition 21 (Semantics of system M) *Once again, we have three rules $R1$, $M1$ and $M2$. These three rules all have a left and right variant, denoted by a subscript l or r, respectively. Their semantic variants are defined as follows:*

- *R1: Application*

$$\begin{array}{lllll} R1_l: & X/Y : \mathsf{Functor}, & Y : \mathsf{Arg} & \Rightarrow & X : \mathsf{Functor\ Arg} \\ R1_r: & Y : \mathsf{Arg}, & Y\backslash X : \mathsf{Functor} & \Rightarrow & X : \mathsf{Functor\ Arg} \end{array}$$

- *M1: Recursion on the range subtype*

$$\begin{array}{lllll} M1_l: & Y\backslash X : \mathsf{Functor}, & Z : \mathsf{SemZ} & \Rightarrow & Y\backslash W : \lambda\mathsf{VarY.\ SemW} \quad \textit{where} \\ & X : \mathsf{Functor\ VarY}, & Z : \mathsf{SemZ} & \Rightarrow & W : \mathsf{SemW} \end{array}$$

$$\begin{array}{lllll} M1_r: & Z : \mathsf{SemZ}, & X/Y : \mathsf{Functor} & \Rightarrow & W/Y : \lambda\mathsf{VarY.\ SemW} \quad \textit{where} \\ & Z : \mathsf{SemZ}, & X : \mathsf{Functor\ VarY} & \Rightarrow & W : \mathsf{SemW} \end{array}$$

- *M2: Recursion on the domain subtype*

$$\begin{array}{lllll} M2_l: & X/Y : \mathsf{Functor}, & Z : \mathsf{SemZ} & \Rightarrow & X/W : \lambda\mathsf{VarW.\ (Functor\ SemY)} \quad \textit{where} \\ & Z : \mathsf{SemZ}, & W : \mathsf{VarW} & \Rightarrow & Y : \mathsf{SemY} \end{array}$$

$$\begin{array}{lllll} M2_r: & Z : \mathsf{SemZ}, & Y\backslash X : \mathsf{Functor} & \Rightarrow & W\backslash X : \lambda\mathsf{VarW.\ (Functor\ SemY)} \quad \textit{where} \\ & W : \mathsf{VarW}, & Z : \mathsf{SemZ} & \Rightarrow & Y : \mathsf{SemY} \end{array}$$

Example To illustrate how the semantic value of the intermediate type is calculated using the above defined rules, we decorate the previous example with the semantic values. The sentence to be interpreted is *"John saw Bill"*, and we have the following lexicon with semantics:

$$\begin{array}{lll} \textbf{John} \vdash \mathbf{j} & : & np \\ \textbf{Bill} \vdash \mathbf{b} & : & np \\ \textbf{saw} \vdash \lambda x.\lambda y.(\textbf{see}\ y\ x) & : & (np\backslash s)/np \end{array}$$

Applying **M** on the first two words *"John saw"* results in the following calculations and corresponding semantic values:

Step	Syntax	Semantics	Rule
1	$np, (np\backslash s)/np \Rightarrow W/np$	$\mathbf{j},\ \lambda x.\lambda y.(\textbf{see}\ y\ x) \Rightarrow \lambda y.\mathsf{SemW}$	$M1_r$
	$np, np\backslash s \Rightarrow W$	$\mathbf{j},\ \lambda y'.(\textbf{see}\ y'\ y) \Rightarrow \mathsf{SemW}$	
	$W = s$	$\mathsf{SemW} = \textbf{see}\ \mathbf{j}\ y$	
2	$np, (np\backslash s)/np \Rightarrow s/np$	$\mathbf{j},\ \lambda x.\lambda y.(\textbf{see}\ y\ x) \Rightarrow \lambda y.(\textbf{see}\ \mathbf{j}\ y)$	Conclusion

We observe that the intermediate type of *"John saw"* equals s/np with the corresponding semantic term $\lambda y.((\textbf{see j})\ y)$. Note that to get the correct semantic term, it is very important to keep in mind which term is the function and which term is the argument.[2]

Continuing with the next intermediate type, which happens to be the result of the sentence. The calculation of the intermediate type of *"John saw"* and *"Bill"* is as follows:

Step	Syntax	Semantics	Rule
1	$s/np, np \Rightarrow s$	$\lambda y.(\textbf{see j}\ y),\ \textbf{b} \Rightarrow \textbf{see j b}$	$R1_l$

As before, this step is a simple application of the argument **b** to the semantic term that corresponds to the intermediate type. The result of the entire sentence is a type s and corresponding semantic term **see j b**. The typelogical derivation is analogously decorated with semantic terms.

In this section we have introduced the system **M** and have observed that it indeed guarantees incrementality. Moreover, it guarantees *strong incrementality* if we restrict the user to always choose the first two input types (may that be an intermediate type and a word, or two words). If we do not impose that restriction, and leave the user free to choose any two types that are next to each other, we still have achieved weak incrementality. Whenever two words are inputted, the procedure of **M** can be used to find an intermediate type and semantic term.[3] This procedure can be iteratively used together with the preceding intermediate types until all input words are processed. Furthermore we can observe that the types of words are not changed, so changing types to accommodate incremental constituents is not necessary. We can also observe that all semantic terms corresponding to intermediate types require an argument to be given. In a broad sense, this can be seen as being the continuation of the sentence where the argument can either be just one word or an entire clause. Strong incremental interpretation is now simply application of the system **M** on the first two types. Every intermediate type with its semantic term is an interpretation of a partial sentence.

However, **M** is based upon **L** and therefore assumes associativity, which is an assumption we did not have in our typelogical grammar of Chapter 2. If we want to be able to interpret wh-phrases, which are based upon the typelogical grammar without associativity, we will need to provide a translation from **NL** to a variant that contains associativity where **M** could be applied. This is possible by taking the associativity of **L** and 'injecting' it in **NL**, as done by Kurtonina and Moortgat [1997]. They achieve a variant of **NL** that contains *controlled associativity* called **NL**$\Diamond$. In the next section we provide a translation to inject controlled associativity into the typelogical grammar of chapter 2 and adapt **M** to accommodate this controlled associativity.

5.3 Incrementally interpreting wh-phrases

So far, we have only interpreted declarative sentences, while the aim of this thesis is achieving incrementality for wh-questions. In Chapter 3 we introduced the wh-type schema originally due to Vermaat [2006]. Vermaat noted that the wh-type schema can also be expressed in the typelogical style only containing slashes. The three different wh-type schemata we introduced

[2]Please note that we have also substituted the original variable y in the term for *'saw'* for the temporary variable y' to make a clear distinction between variables.

[3]If the rules do not extend far enough, types can also be combined with a $\bullet$ as we will see later. We will call this *composition*.

(ex-situ with the gap on a left branch, ex-situ with the gap on a right branch and in-situ) can be rewritten as follows:

$$WH^{l}_{ex}(A,B,C) = C/(A\backslash B)$$
$$WH^{r}_{ex}(A,B,C) = C/(B/A)$$
$$WH_{in}(A,B,C) = \Diamond\Box(C/B) \bullet A$$

These rewritten types do lend themselves for the system **M**, but as we have mentioned in the previous chapter, the system **M** is based upon **L** and our typelogical grammar in which the wh-type schemata have been defined is based upon **NL**. Kurtonina and Moortgat [1997] proposed a translation to embed the structural flexibility (associativity) of **L** in **NL**. The result is **NL**$\Diamond$ where structural operators are used to control associativity. To incrementally interpret the wh-phrases introduced in Chapter 3, we are looking for a version of **M** that accommodates the structural flexibility (associativity) of **L** but is written in terms of **NL**. The translation is as follows:[4]

$$\begin{aligned}
p^{\sharp} &= p \\
(A \circ_1 B)^{\sharp} &= \blacklozenge(A^{\sharp} \circ_0 B^{\sharp}) \\
(A/_1 B)^{\sharp} &= \Box A^{\sharp}/_0 B^{\sharp} \\
(B\backslash_1 A)^{\sharp} &= B^{\sharp}\backslash_0 \Box A^{\sharp} \\
(\Diamond_1 A)^{\sharp} &= \blacklozenge_0 A^{\sharp} \\
(\Box_1 A)^{\sharp} &= \Box_0 A^{\sharp}
\end{aligned}$$

Note that we translate a $\Diamond$ to a $\blacklozenge$ to be in line with the typelogical grammar presented in Chapter 2. The rules of **M** make use of the general associativity rule that is not decorated with structural operators. So, our next challenge is to translate the rules of **M** to rules that accommodate controlled associativity. However, as our goal is incremental interpretation in which bracketing is of the form $(((A_1 \circ A_2) \circ A_3) \circ ... \circ A_n)$, where A_i is a type of the grammar, we will only need one direction of associativity. Moreover, we only need to translate the rules from **M** that use that one direction of associativity. We use the following structural rule of associativity. In this rule the structure is rebracketed from right to left:

$$\frac{\Gamma[\blacklozenge(\Delta_1 \circ \blacklozenge(\Delta_2 \circ \Delta_3))] \vdash C}{\Gamma[\blacklozenge(\blacklozenge(\Delta_1 \circ \Delta_2) \circ \Delta_3)] \vdash C}\ [Ass_{\blacklozenge}]$$

In the remainder of this section we will provide the original derivation of the rules of **M** next to the translated derivation for our new version of **M**. We only provide these derivations for rules that make use of associativity in the direction discussed above, which are the rules $R1_l$, $R1_r$, $M1_r$ and $M2_l$. After that we will provide a new variant of the rule $M2_l$ that we add to the system to accommodate the reordering of structures as that is necessary for ex-situ wh-phrases with gaps on right branches.[5] We will present the derivations per discussed rule and provide the rule of system **M** that is derived below the derivation.

[4] The subscript 1 and 0 denote operators from the **L** and **NL**, respectively.

[5] As we have seen in Chapter 3, a structural postulate is necessary to move the gap to the correct position for ex-situ wh-phrases with the gap on a right branch.

Application rules
For the application rules $R1_l$ and $R1_r$ the translation is straightforward. Even though they do not make use of associativity, we will need these application rules in our new system as a base case for recursion.[6]

$R1_l$:

$$\dfrac{\dfrac{}{X/Y \vdash X/Y}\ Ax \quad \dfrac{}{Y \vdash Y}\ Ax}{X/Y \circ Y \vdash X}\ [/E] \qquad \dfrac{\dfrac{\dfrac{}{\Box X/Y \vdash \Box X/Y}\ Ax \quad \dfrac{}{Y \vdash Y}\ Ax}{\Box X/Y \circ Y \vdash \Box X}\ [/E]}{\blacklozenge(\Box X/Y \circ Y) \vdash X}\ [\Box E]$$

$$R1_l : X/Y \circ Y \Rightarrow X \qquad R1_l : \blacklozenge(\Box X/Y \circ Y) \Rightarrow X$$

$R1_r$:

$$\dfrac{\dfrac{}{Y \vdash Y}\ Ax \quad \dfrac{}{Y\backslash X \vdash Y\backslash X}\ Ax}{Y \circ Y\backslash X \vdash X}\ [\backslash E] \qquad \dfrac{\dfrac{\dfrac{}{Y \vdash Y}\ Ax \quad \dfrac{}{\Box X\backslash Y \vdash \Box X\backslash Y}\ Ax}{Y \circ Y\backslash \Box X \vdash \Box X}\ [\backslash E]}{\blacklozenge(Y \circ Y\backslash \Box X) \vdash X}\ [\Box E]$$

$$R1_r : Y \circ Y\backslash X \Rightarrow X \qquad R1_r : \blacklozenge(Y \circ Y\backslash \Box X) \Rightarrow X$$

Recursion on the range subtype
For the rule $M1_r$, associativity comes into play and we'll replace it with the structurally controlled associativity accordingly. Note that we only present $M1_r$, since this is the only necessary rule to achieve incrementality.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Assumption}{Z \circ X \vdash W}\ [/I]}{Z \vdash W/X} \quad \dfrac{\dfrac{}{X/Y \vdash X/Y}\ Ax \quad \dfrac{}{Y \vdash Y}\ Ax}{X/Y \circ Y \vdash X}\ [/E]}{Z \circ (X/Y \circ Y) \vdash W}\ [/E]}{(Z \circ X/Y) \circ Y \vdash W}\ [Ass]}{Z \circ X/Y \vdash W/Y}\ [/I]$$

$$M1_r : \quad Z \circ X/Y \Rightarrow W/Y \text{ where } Z \circ X \Rightarrow W$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Assumption}{\blacklozenge(Z \circ X) \vdash W}\ [\Box I]}{Z \circ X \vdash \Box W}\ [/I]}{Z \vdash \Box W/X} \quad \dfrac{\dfrac{\dfrac{}{\Box X/Y \vdash \Box X/Y}\ Ax \quad \dfrac{}{Y \vdash Y}\ Ax}{\Box X/Y \circ Y \vdash \Box X}\ [/E]}{\blacklozenge(\Box X/Y \circ Y) \vdash X}\ [\Box E]}{Z \circ \blacklozenge(\Box X/Y \circ Y) \vdash \Box W}\ [/E]}{\blacklozenge(Z \circ \blacklozenge(\Box X/Y \circ Y)) \vdash W}\ [\Box E]}{\blacklozenge(\blacklozenge(Z \circ \Box X/Y) \circ Y) \vdash W}\ [Ass_{\blacklozenge}]}{\blacklozenge(Z \circ \Box X/Y) \circ Y \vdash \Box W}\ [\Box I]}{\blacklozenge(Z \circ \Box X/Y) \vdash \Box W/Y}\ [/I]$$

$$M1_r : \quad \blacklozenge(Z \circ \Box X/Y) \Rightarrow \Box W/Y \text{ where } \blacklozenge(Z \circ X) \Rightarrow W$$

[6] In the rules of **M** we replace the ',' by a $\circ$ to clarify the correspondence between the derivation and the rule. This means the same, as the comma was just a notational difference.

Recursion on the domain subtype
For the $M2$ rules, only the left variant is used for incremental interpretation. We replace the use of general associativity with the structurally controlled associativity accordingly.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{X/Y \vdash X/Y}\,Ax \quad \dfrac{}{Z \circ W \vdash Y}\,Assumption}{X/Y \circ (Z \circ W) \vdash X}\,[/E]}{(X/Y \circ Z) \circ W \vdash X}\,[Ass]}{X/Y \circ Z \vdash X/W}\,[/I]$$

$M2_l$: $X/Y \circ Z \Rightarrow X/W$ where $Z \circ W \Rightarrow Y$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Box X/Y \vdash \Box X/Y}\,Ax \quad \dfrac{}{\blacklozenge(Z \circ W) \vdash Y}\,Assumption}{\Box X/Y \circ \blacklozenge(Z \circ W) \vdash \Box X}\,[/E]}{\blacklozenge(\Box X/Y \circ \blacklozenge(Z \circ W)) \vdash X}\,[\Box E]}{\blacklozenge(\blacklozenge(\Box X/Y \circ Z) \circ W) \vdash X}\,[Ass_{\blacklozenge}]}{\blacklozenge(\Box X/Y \circ Z) \circ W \vdash \Box X}\,[\Box I]}{\blacklozenge(\Box X/Y \circ Z) \vdash \Box X/W}\,[/I]$$

$M2_l$: $\blacklozenge(\Box X/Y \circ Z) \Rightarrow \Box X/W$ where $\blacklozenge(Z \circ W) \Rightarrow Y$

The new system M$\blacklozenge$
At this point we have established the rules of our new version of **M**, called **M$\blacklozenge$**. With these rules we can find intermediate types of wh-phrases. However, we will see that it is not complete yet, as it is not able to find an intermediate type for wh-phrases where the gap is decorated with structural operators, such as **whom** :: $\mathrm{WH}^r_{ex}(\Diamond\Box np, q, wh)$. Later, we will see how we can incorporate such types in our system.

Considering the semantic terms calculated by these new rules, nothing has changed. The translation and thereby all differences between the rules of **M$\blacklozenge$** and **M** is purely syntactical. All rules of **M$\blacklozenge$** that we will use will look as follows when semantic terms are decorated over them:

$$\begin{array}{lllll}
R1_l\text{:} & \blacklozenge(\Box X/Y : \mathsf{Functor} \circ Y : \mathsf{Arg}) & \Rightarrow X : \mathsf{Functor\ Arg} & & \\
R1_r\text{:} & \blacklozenge(Y : \mathsf{Arg} \circ Y\backslash\Box X : \mathsf{Functor}) & \Rightarrow X : \mathsf{Functor\ Arg} & & \\
 & & & & \\
M1_r\text{:} & \blacklozenge(Z : \mathsf{SemZ} \circ \Box X/Y : \mathsf{Functor}) & \Rightarrow \Box W/Y : \lambda\mathsf{VarY.\ SemW} & \text{where} & \\
 & \blacklozenge(Z : \mathsf{SemZ} \circ X : \mathsf{Functor\ VarY}) & \Rightarrow W : \mathsf{SemW} & & \\
 & & & & \\
M2_l\text{:} & \blacklozenge(\Box X/Y : \mathsf{Functor} \circ Z : \mathsf{SemZ}) & \Rightarrow \Box X/W : \lambda\mathsf{VarW.\ (Functor\ SemY)} & \text{where} & \\
 & \blacklozenge(Z : \mathsf{SemZ} \circ W : \mathsf{VarW}) & \Rightarrow Y : \mathsf{SemY} & &
\end{array}$$

Something else that this system will not be able to account for is in-situ wh-phrases. As their syntactic type is $\Diamond\Box(C/B) \bullet A$, and we have no rules in **M$\blacklozenge$** that account for a product operator as main operator of a type, and due to time constraints, we must sadly leave in-situ wh-phrases out of scope of this thesis. We do see the incremental interpretation of in-situ wh-phrases being modelled within **M$\blacklozenge$** in further research. One might, for example, add a new rule to the system that does account for a product operator as main operator of a type if one knows what the intermediate type of such a type and a second type might be. Nevertheless, **M$\blacklozenge$** as we have presented so far, is a start for incremental interpretation of wh-questions. In the remainder of this chapter we present examples of incremental interpretation for ex-situ wh-phrases. Eventually an algorithm will be presented that makes use of the system **M$\blacklozenge$** internally, to achieve incremental interpretation.

5.3.1 Ex-situ

As we have seen before, ex-situ wh-phrases are divided into two variants. One where the gap hypothesis is located on a left branch, WH^l_{ex}, and one where the gap hypothesis is located on a right branch, WH^r_{ex}. **M♦** as we have presented it in the previous section as ready to interpret the first variant of ex-situ wh-phrases. In this section we'll provide an example of the incremental interpretation of wh-phrases of this variant. However, as stated a few times earlier, for the second variant of ex-situ wh-phrases the gap is decorated with structural control operators in order to be able to displace it to the correct position for the insertion of the wh-phrase. **M♦** as we have presented it so far, is not able to accommodate those structural control operators. In this section we will add an extra rule $M2\blacklozenge$ (based on $M2_l$) to accommodate gaps decorated with structural control operators. We will need a reordering postulate introduced in chapter 3 in order to derive this extra rule.

Example WH^l_{ex}

For the example of incremental interpretation of ex-situ wh-phrases with the gap on a left branch, we will apply **M♦** to the wh-question *"Who saw Bill?"*. To do so, we use the following lexicon:

$$\begin{array}{ll} \textbf{who} \vdash \lambda P.\lambda Q.\ (Q\ P) & : wh/(np\backslash s) \\ \textbf{saw} \vdash \lambda x.\lambda y.\ (\textbf{see}\ y\ x) & : (np\backslash s)/np \\ \textbf{Bill} \vdash \textbf{b} & : np \end{array}$$

To interpret these wh-questions the first step to undertake is translating these types to types that are fit as input for **M♦**. Applying our translation results in the following lexicon:

$$\begin{array}{ll} \textbf{who} \vdash \lambda P.\lambda Q.\ (Q\ P) & : \Box wh/(np\backslash\Box s) \\ \textbf{saw} \vdash \lambda x.\lambda y.\ (\textbf{see}\ y\ x) & : \Box(np\backslash\Box s)/np \\ \textbf{Bill} \vdash \textbf{b} & : np \end{array}$$

We start applying **M♦** to the first two words of the question, *'who'* and *'saw'*. In order to be able to apply **M♦** to their translated types, we have to combine them with a structural product and wrap them in a ♦. This will result in the following calculations:

Step	Calculation	Rule
1	$\blacklozenge(\Box wh/(np\backslash\Box s) \circ \Box(np\backslash\Box s)/np) \Rightarrow \Box wh/W$	$M2_l$
	$\blacklozenge(\Box(np\backslash\Box s)/np \circ W) \Rightarrow np\backslash\Box s$	$R1_l$
	$W = np$	
2	$\blacklozenge(\Box wh/(np\backslash\Box s) \circ \Box(np\backslash\Box s)/np) \Rightarrow \Box wh/np$	*Conclusion*

And so we have found the intermediate type $\Box wh/np$ for the phrase *"Who saw"* with corresponding semantic term $\lambda w.\lambda Q.\ (Q\ (\lambda y.\ (\textbf{see}\ y\ w))$. Continuing with the third and last word of the sentence *"Bill"*, we combine the intermediate type of the preceding phrase ($\Box wh/np$) with the translated type of *"Bill"* with a structural product and once again, wrap it in a ♦. **M♦** then returns the following calculations:

Step	Calculation	Rule
1	$\blacklozenge(\Box wh/np \circ np) \Rightarrow wh$	$R1_l$

The resulting intermediate type and type of the entire sentence is wh with corresponding semantic term $\lambda Q.\ (Q\ (\lambda y.\ (\mathbf{see}\ y\ \mathbf{b}))$.

Example WH^r_{ex}

For the example of incremental interpration of ex-situ wh-phrases with the gap on a right branch, we will apply **M♦** to the wh-question *"Whom did Bill see?"*. We will observe that with the current state of **M♦** we are not able to derive this sentence completely. But before we get there, we'll proceed as usual and use the following lexicon:

$$\begin{array}{lll} \mathbf{whom} & \vdash \lambda P.\lambda Q.\ (Q\ P) & : wh/(q/\Diamond\Box np) \\ \mathbf{did} & \vdash \lambda P.\lambda x.\ (P\ x) & : (q/inf)/np \\ \mathbf{Bill} & \vdash \mathbf{b} & : np \\ \mathbf{see} & \vdash \lambda x.\ (\mathbf{see}\ x) & : inf/np \end{array}$$

Similar to the previous example, we have to translate the types in this lexicon to types that are fit to be used with **M♦**. Applying our translation results in the following lexicon:

$$\begin{array}{lll} \mathbf{whom} & \vdash \lambda P.\lambda Q.\ (Q\ P) & : \Box wh/(\Box q/\blacklozenge\Box np) \\ \mathbf{did} & \vdash \lambda P.\lambda x.\ (P\ x) & : \Box(\Box q/inf)/np \\ \mathbf{Bill} & \vdash \mathbf{b} & : np \\ \mathbf{see} & \vdash \lambda x.\ (\mathbf{see}\ x) & : \Box inf/np \end{array}$$

Applying **M♦** to the first two words *'whom'* and *'did'*, we get the following calculations:[7]

Step	Calculation	Rule
1	$\blacklozenge(\Box wh/(\Box q/\blacklozenge\Box np) \circ \Box(\Box q/inf)/np \Rightarrow \Box W/np$	$M1_r$
	$\blacklozenge(\Box wh/(\Box q/\blacklozenge\Box np) \circ \Box q/inf) \Rightarrow W$	Rewrite
2	$\blacklozenge(\Box wh/(\Box q/\blacklozenge\Box np) \circ \Box q/inf) \Rightarrow \Box wh/V$	$M2_l$
	$\blacklozenge(\Box q/inf \circ V) \Rightarrow \Box q/\blacklozenge\Box np$	
3	$\blacklozenge(\Box q/inf \circ V) \Rightarrow \Box q/\blacklozenge\Box np$	$M2_l$
	$\blacklozenge(V \circ \blacklozenge\Box np) \Rightarrow inf$	
	$V = \Box inf/\blacklozenge\Box np$	
	$W = \Box wh/(\Box inf/\blacklozenge\Box np)$	
	$I = \Box(\Box wh/(\Box inf/\blacklozenge\Box np))/np$	

For the words *'whom'* and *'did'* we have found the intermediate type $\Box(\Box wh/(\Box inf/\blacklozenge\Box np))/np$ with corresponding semantic term $\lambda y.\lambda w.\lambda Q.\ (Q\ (\lambda u.\ w\ y\ u))$. Continuing with the next word in the question, *'Bill'*, we find the following calculations:

Step	Calculation	Rule
1	$\blacklozenge(\Box(\Box wh/(\Box inf/\blacklozenge\Box np))/np \circ np) \Rightarrow \Box wh/(\Box inf/\blacklozenge\Box np)$	$R1_l$

By simple application we have found the intermediate type $\Box wh/(\Box inf/\blacklozenge\Box np)$ with corresponding semantic term $\lambda w.\lambda Q.\ (Q\ (\lambda u.\ w\ \mathbf{b}\ u))$. For the last word of the question, *'see'*, we now observe the flaw in **M♦**. The gap decorated with structural operators in the type of *'whom'*, is still decorated with structural operators (although translated) within the intermediate type, and

[7] We use a rewrite step to clarify how we get to the eventual intermediate type.

we do not have a rule available in **M♦** that is applicable in this case. We propose a new rule, based on application to deal with this case. And as we know the gap decorated with structural operators will only follow from type of the wh-phrase, it is also known that the gap will only occur in the lefthand type of the structural product. The following rule is based on $R1_l$, but we break the lefthand type up to make the structural operators explicit. Semantic terms once again do not change, since the decoration of structural operators is purely syntactical. Therefore, we will omit the semantic terms in the following derivations. Due to lack of space we write the derivation in **NL**$\Diamond$ below the derivation in **L**.

$R2$:

$$\dfrac{\dfrac{}{X/(Y1/\Diamond\Box Y2) \vdash X/(Y1/\Diamond\Box Y2)}\ Ax \qquad \dfrac{\dfrac{\dfrac{}{Y1/Y2 \vdash Y1/Y2}\ Ax \quad \dfrac{\dfrac{}{\Box Y2 \vdash \Box Y2}\ Ax}{\Diamond\Box Y2 \vdash Y2}\ [\Box E]}{Y1/Y2 \circ \Diamond\Box Y2 \vdash Y1}\ [/E]}{Y1/Y2 \vdash Y1/\Diamond\Box Y2}\ [/I]}{X/(Y1/\Diamond\Box Y2) \circ Y1/Y2 \vdash X}\ [/E]$$

$$R2 : X/(Y1/\Diamond\Box Y2) \circ Y1/Y2 \Rightarrow X$$

$$\dfrac{\dfrac{}{\Box X/(\Box Y1/\blacklozenge\Box Y2) \vdash \Box X/(\Box Y1/\blacklozenge\Box Y2)}\ Ax \qquad \dfrac{\dfrac{\dfrac{}{\Box Y1/Y2 \vdash \Box Y1/Y2}\ Ax \quad \dfrac{\dfrac{}{\Box Y2 \vdash \Box Y2}\ Ax}{\blacklozenge\Box Y2 \vdash Y2}\ [\Box E]}{\Box Y1/Y2 \circ \blacklozenge\Box Y2 \vdash \Box Y1}\ [/E]}{\Box Y1/Y2 \vdash \Box Y1/\blacklozenge\Box Y2}\ [/I]}{\dfrac{\Box X/(\Box Y1/\blacklozenge\Box Y2) \circ \Box Y1/Y2 \vdash \Box X}{\blacklozenge(\Box X/(\Box Y1/\blacklozenge\Box Y2) \circ \Box Y1/Y2) \vdash X}\ [\Box E]}\ [/E]$$

$$R2 : \blacklozenge(\Box X/(\Box Y1/\blacklozenge\Box Y2) \circ \Box Y1/Y2) \Rightarrow X$$

The addition of $R2$ to **M♦** ensures that we can now interpret the last word through the following calculations:

Step	Calculation	Rule
1	$\Box wh/(\Box inf/\blacklozenge\Box np) \circ \Box inf/np) \Rightarrow wh$	$R2$

We have now incrementally interpreted a wh-question with an ex-situ wh-phrase with the gap on a right branch! The final intermediate type, or the type for the entire wh-question, is wh with corresponding semantic term $\lambda Q.\ (Q\ (\lambda u.\ \textbf{see b}\ u))$.

However, what if the word *'see'* was not the end of the question? What if we wanted to interpret the wh-question *"What did Bill put on the table?"*? In this case the gap created by the wh-phrase *'what'* would follow the word *'put'* and is reflected by the np in its type $(inf/pp)/np$. We would not be able to apply the rule $R2$ we just added to the system **M♦**. To interpret such questions, we add another rule to the system **M♦** which will be similar to the recursive case $M2_l$. This new rule will make use of a controlled variant of the structural reordering postulates defined by Vermaat [2006] to move the gap to its desired position, similar to the controlled variant of the rule of associativity:

$$\dfrac{\blacklozenge(\blacklozenge(\Delta_1 \circ \blacklozenge\Delta_3) \circ \Delta_2) \vdash \Gamma}{\blacklozenge(\blacklozenge(\Delta_1 \circ \Delta_2) \circ \blacklozenge\Delta_3) \vdash \Gamma}\ [Pr2_{\blacklozenge}]$$

Once again, if we make the structural operators on the gap type explicit in $M2_l$, the new rule is derived in **L** and **NL**$\Diamond$ as follows:

$M2\blacklozenge$:

$$\dfrac{\dfrac{}{X/(Y1/\Diamond\Box Y2) \vdash X/(Y1/\Diamond\Box Y2)}\,Ax \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Assumption}{Z \circ Y2 \vdash Y1/W}}{Z \vdash (Y1/W)/Y2}\,[/I] \quad \dfrac{\dfrac{}{\Box Y2 \vdash \Box Y2}\,Ax}{\Diamond\Box Y2 \vdash Y2}\,[\Box E]}{Z \circ \Diamond\Box Y2 \vdash Y1/W}\,[/E] \quad \dfrac{}{W \vdash W}\,Ax}{(Z \circ \Diamond\Box Y2) \circ W \vdash Y1}\,[/E]}{(Z \circ W) \circ \Diamond\Box Y2 \vdash Y1}\,[Pr2]}{Z \circ W \vdash Y1/\Diamond\Box Y2}\,[/I]}{}}{\dfrac{\dfrac{X/(Y1/\Diamond\Box Y2) \circ (Z \circ W) \vdash X}{(X/(Y1/\Diamond\Box Y2) \circ Z) \circ W \vdash X}\,[Ass]}{X/(Y1/\Diamond\Box Y2) \circ Z \vdash X/W}\,[/I]}\,[/E]$$

$M2\blacklozenge$: $X/(Y1/\Diamond\Box Y2) \circ Z \Rightarrow X/W$ where
$Z \circ Y2 \Rightarrow Y1/W$

$$\dfrac{\dfrac{}{\Box X/(\Box Y1/\blacklozenge\Box Y2) \vdash \Box X/(\Box Y1/\blacklozenge\Box Y2)}\,Ax \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{Assumption}{\blacklozenge(Z \circ Y2) \vdash \Box Y1/W}}{Z \circ Y2 \vdash \Box(\Box Y1/W)}\,[\Box I]}{Z \vdash \Box(\Box Y1/W)/Y2}\,[/I] \quad \dfrac{\dfrac{}{\Box Y2 \vdash \Box Y2}\,Ax}{\blacklozenge\Box Y2 \vdash Y2}\,[\Box E]}{Z \circ \blacklozenge\Box Y2 \vdash \Box(\Box Y1/W)}\,[/E]}{\blacklozenge(Z \circ \blacklozenge\Box Y2) \vdash \Box Y1/W}\,[\Box E] \quad \dfrac{}{W \vdash W}\,Ax}{\blacklozenge(Z \circ \blacklozenge\Box Y2) \circ W \vdash \Box Y1}\,[/E]}{\blacklozenge(\blacklozenge(Z \circ \blacklozenge\Box Y2) \circ W) \vdash Y1}\,[\Box I]}{\blacklozenge(\blacklozenge(Z \circ W) \circ \blacklozenge\Box Y2) \vdash Y1}\,[Pr2_{\blacklozenge}]}{\blacklozenge(Z \circ W) \circ \blacklozenge\Box Y2 \vdash \Box Y1}\,[\Box E]}{\blacklozenge(Z \circ W) \vdash \Box Y1/\blacklozenge\Box Y2}\,[/I]}{\dfrac{\dfrac{\dfrac{\dfrac{\Box X/(\Box Y1/\blacklozenge\Box Y2) \circ \blacklozenge(Z \circ W) \vdash \Box X}{\blacklozenge(\Box X/(\Box Y1/\blacklozenge\Box Y2) \circ \blacklozenge(Z \circ W)) \vdash X}\,[\Box E]}{\blacklozenge(\blacklozenge(\Box X/(\Box Y1/\blacklozenge\Box Y2) \circ Z) \circ W) \vdash X}\,[Ass_{\blacklozenge}]}{\blacklozenge(\Box X/(\Box Y1/\blacklozenge\Box Y2) \circ Z) \circ W \vdash \Box X}\,[\Box I]}{\blacklozenge(\Box X/(\Box Y1/\blacklozenge\Box Y2) \circ Z) \vdash \Box X/W}\,[/I]}\,[/E]$$

$M2\blacklozenge$: $\blacklozenge(\Box X/(\Box Y1/\blacklozenge\Box Y2) \circ Z) \Rightarrow \Box X/W$ where
$\blacklozenge(Z \circ Y2) \Rightarrow \Box Y1/W$

The rule $M2\blacklozenge$ with semantic terms decorated over it, looks as follow:

$M2\blacklozenge$: $\blacklozenge(\Box X/(\Box Y1/\blacklozenge\Box Y2) : \mathsf{Functor} \circ Z : \mathsf{SemZ}) \Rightarrow \Box X/W : \lambda\mathsf{VarW}.\ (\mathsf{Functor}\ (\lambda\mathsf{VarY2}.\ \mathsf{SemY1}))$ where
$\blacklozenge(Z : \mathsf{SemZ} \circ Y2 : \mathsf{VarY2}) \Rightarrow \Box Y1/W : \lambda\mathsf{VarW}.\ \mathsf{SemY1}$

With this new rule, $M2\blacklozenge$, we can now interpret a wh-question such as *"What did Bill put on the*

table?" incrementally. Let us use the following translated lexicon:[8]

what	$\vdash \lambda P.\lambda Q.\ (Q\ P)$	$: \Box wh/(\Box q/\blacklozenge\Box np)$	**on**	$\vdash \lambda x.(\mathbf{on}\ x)$	$: pp/np$
did	$\vdash \lambda P.\lambda x.\ (P\ x)$	$: \Box(\Box q/inf)/np$	**the**	$\vdash \lambda Q.\iota z.(Q\ z)$	$: np/n$
Bill	$\vdash \mathbf{b}$	$: np$	**table**	$\vdash \mathbf{table}$	$: n$
put	$\vdash \lambda x.\lambda y.\lambda z.\ (\mathbf{put}\ z\ x\ y)$	$: \Box(\Box inf/pp)/np$			

Suppose we have found intermediate type $\Box wh/(\Box inf/\blacklozenge\Box np)$ for the phrase *"What did Bill"*.[9] Applying our new rule $M2\blacklozenge$ to this intermediate type and the next word *'put'* results in the following calculations:

Step	Calculation	Rule
1	$\Box wh/(\Box inf/\blacklozenge\Box np) \circ \Box(\Box inf/pp)/np) \Rightarrow wh/W$	$M2\blacklozenge$
	$\blacklozenge(\Box(\Box inf/pp)/np \circ np) \Rightarrow \Box inf/W$	
	$W = pp$	
2	$\Box wh/(\Box inf/\blacklozenge\Box np) \circ \Box(\Box inf/pp)/np) \Rightarrow wh/pp$	*Conclusion*

We have found the intermediate type wh/pp for the phrase *"What did Bill put"* with semantic term $\lambda w.\lambda Q.\ (Q\ (\lambda u.\ (\mathbf{put}\ \mathbf{b}\ u\ w)))$. The remainder of the wh-question is incrementally interpreted by application as shown before.

The entire system M♦

At this point, we have established our new system. **M♦** is a variant of the original system **M** due to Moortgat [1988], but instead of relying on general associativity it relies on structurally controlled variants of associativity and reordering. We have provided a few examples that show that the system **M♦** is capable of calculating an intermediate type of two types, even if the first of those two types is a wh-phrase. It is not, however, capable of calculating an intermediate type when that wh-phrase is of the in-situ variant. The rules that constitute the system **M♦** do not account for a product operator as a main connective of a type, whereas the types of in-situ wh-phrases do have a product operator as main connective.

Here we repeat all rules established for **M♦** before moving on to describing an algorithm for incremental interpretation:

[8] Read the semantic term of *'put'* as: z put x at location y

[9] Note that this intermediate type is equal to the intermediate type for *"Whom did Bill"*.

$R1_l$: $\blacklozenge(\Box X/Y$: Functor $\circ\ Y$: Arg) $\Rightarrow X$: Functor Arg

$R1_r$: $\blacklozenge(Y$: Arg $\circ\ Y\backslash\Box X$: Functor) $\Rightarrow X$: Functor Arg

$R2$: $\blacklozenge(\Box X/(\Box Y1/\blacklozenge\Box Y2)$: Functor $\circ\ \Box Y1/Y2$: Arg) $\Rightarrow X$: Functor Arg

$M1_r$: $\blacklozenge(Z$: SemZ $\circ\ \Box X/Y$: Functor) $\Rightarrow \Box W/Y : \lambda$VarY. SemW where
$\blacklozenge(Z$: SemZ $\circ\ X$: Functor VarY) $\Rightarrow W$: SemW

$M2_l$: $\blacklozenge(\Box X/Y$: Functor $\circ\ Z$: SemZ) $\Rightarrow \Box X/W : \lambda$VarW. (Functor SemY) where
$\blacklozenge(Z$: SemZ $\circ\ W$: VarW) $\Rightarrow Y$: SemY

$M2\blacklozenge$: $\blacklozenge(\Box X/(\Box Y1/\blacklozenge\Box Y2)$: Functor $\circ\ Z$: SemZ) $\Rightarrow \Box X/W : \lambda$VarW. (Functor ($\lambda$VarY2. SemY1)) where
$\blacklozenge(Z$: SemZ $\circ\ Y2$: VarY2) $\Rightarrow \Box Y1/W : \lambda$VarW. SemY1

5.4 Algorithm for incremental interpretation

Now we have established a system that is able to calculate intermediate types for ex-situ wh-phrases, how do we put it to work? The reader may have noticed that every time we started interpretation of a wh-question, the lexicon was translated from types in **NL** to types in **NL**$\Diamond$. This is step 1: Translating the lexicon. For example, the types np and $(np\backslash s)/np$ are translated to np and $\Box(np\backslash\Box s)/np$.

After the translation had been made, every time we chose two phrases to apply the system **M**$\blacklozenge$ to, we combined them with a structural operator $\circ$ and wrapped the result of that product in a $\blacklozenge$. This is step 2: Preparing the types for application of system **M**$\blacklozenge$. For example, the types np and $\Box(np\backslash\Box s)/np$ are prepared to be the input structure $\blacklozenge(np \circ \Box(np\backslash\Box s)/np)$.

At this point, the phrases that we want to know the intermediate type of are ready for the application of **M**$\blacklozenge$. So naturally, step 3 is: Apply the system **M**$\blacklozenge$ to the result of step 2. For example, the intermediate type calculated from the input structure $\blacklozenge(np \circ \Box(np\backslash\Box s)/np)$ will be the type $\Box s/np$.

Repeating step 2 and step 3 iteratively with the previous intermediate type and the next input type until all words are interpreted is the incremental interpretation of sentences. This algorithm may be applied to wh-questions when their wh-type schema is translated to a typelogical type containing only slashes, but also to declaratives or questions. The only premise is that the types that the input types for the algorithm should not contain the product operator.

M$\blacklozenge$, when executed, will try to find any rule that pattern matches to the input structure and apply itself recursively until an intermediate type has been found. Whenever it does not find an intermediate type by applying a rule, it is allowed to backtrack and search for more rules that can be applied. This allows **M**$\blacklozenge$ to have multiple paths towards the intermediate type and also results in multiple intermediate types that can be chosen for the next input. Therefore **M**$\blacklozenge$ is non-deterministic in nature. At some point during the calculation of an intermediate type, it may have multiple rules that it can choose to apply to the input and that may result in different

intermediate results for one and the same input. Moreover, because of this non-determinism, giving the same input twice to **M♦** does not necessarily result in the same intermediate type. However, when applying the same rules of **M♦** in the same order to the same input, does result in the same intermediate type.

Eventually, **M♦** will return an intermediate type and this intermediate type can be used in a typelogical derivation using a Cut-rule. Repeating this process on every next word in the wh-question and connecting that word to the intermediate type corresponding to the preceding phrase until no words are left as input, is the incremental interpretation of wh-questions. So for example, say we receive the types $A_1, ..., A_n$ incrementally by a speaker who utters a question to us and we want to interpret these types while the speaker is still speaking.[10] We may use the following program in pseudocode for this goal:

Data: Lexicon, a stack S of input types $A_1, ..., A_n$
Result: An output type B
translateLexicon;
while *S contains 2 or more types* **do**
  pop first two types on S;
  combine types with structural product;
  wrap types in ♦;
  apply **M♦** to wrapped type and return intermediate type;
  push intermediate type to S;
end

A typelogical derivation such as the following illustrates the use of the Cut-rule and the intermediate type, and provides an intuition about the incrementality achieved by this algorithm:[11]

$$\dfrac{\dfrac{\vdots}{A_1 \circ A_2 \vdash B_1} \quad \dfrac{\dfrac{\vdots}{B_1 \circ A_3 \vdash B_2} \quad \dfrac{\vdots}{B_2 \circ A_4 \circ ... \circ A_n \vdash R}\,[Cut]}{B_1 \circ A_3 \circ ... \circ A_n \vdash R}\,[Cut]}{A_1 \circ ... \circ A_n \vdash R}$$

In this typelogical derivation we can observe that every first two input types of the antecedent of the sequent (the stack) are combined to an intermediate type when the Cut-rule is applied. This continues until there are two types left in the antecedent of the sequent and the derivation can be finished as usual. The purpose of using **M♦** within this algorithm (and therefore within this derivation) is to calculate the intermediate types $B_1, ..., B_m$ efficiently, since **M♦** is able to bypass the conventional constituency of a sentence. As generating such a derivation is possible when using the algorithm, it is confirmed that the sentence is also grammatically correct. We may use this algorithm not only for the incremental interpretation wh-questions, but also for regular questions or declaratives, as it is possible to translate such types and apply **M♦** to them as well. This makes the use case of the algorithm more general then just the incremental interpretation of wh-questions.

The semantic term for the intermediate type follows from the rules of **M♦** automatically, as we have shown before and can also be returned by the above algorithm. It is a process of substitution

[10] The parentheses are omitted in this structure as we assume the input to be delivered incrementally. So the structure with parentheses would look like $(((A_1 \circ A_2) \circ A_3) \circ ... \circ A_n)$.

[11] This is a very simplified picture of the situation, just for illustrational purposes!

and unification, until the correct term has been derived. Within the algorithm, after every time the system **M♦** is applied to the input, a semantic term will correspond to the intermediate type that is returned. This semantic term could also be return by **M♦** and used within the algorithm for further analysis. For example, when the user of the algorithm wants its implementation to predict what the rest of the sentence may be. When the intermediate type that has been found by **M♦** is not the sentence type (or (wh-)question type), the semantic term will always require an argument to be filled in. This argument could be called the continuation of the sentence. The semantic term of the intermediate type is the term corresponding to a partial phrase of the sentence. And seeing as it contains a continuation, this term can be used to reason over future input. Moreover, there exists information in the typing of the term. As stated before, the semantic term corresponding to a type s/np requires an argument of type e for entities. So from this term it can already be derived that the next word that may be given as input is an entity. Reasoning about these semantic terms may give knowledge about how the sentence will end or, in the case of wh-questions, may give an indication as towards the answer to the question. The continuation within the partial semantic term plays a big part in acquiring this knowledge.

5.5 Conclusion

To conclude this chapter, a few remarks have to be made on the presented system **M♦** and algorithm. As noted before, the presented algorithm is able to interpret wh-questions incrementally, as the system **M♦** is able to calculate the intermediate type of an ex-situ wh-phrase and another word. However, the algorithm is not able to interpret all wh-questions. For example, the incremental interpretation of multiple wh-questions is not possible, as multiple wh-questions always contain an in-situ wh-phrase and **M♦** is not capable of calculating an intermediate type of an in-situ wh-phrase and another word. This is, as stated before, due to the fact that **M♦** does not encapsulate a rule that accommodates product operators as main connectives of a type, whereas the type of an in-situ wh-phrase does have a product operator as main connective. Solving this problem would require a new rule for **M♦** that accommodates a product operator as main connective, but it raises the question what the intermediate type should be of a product type and another type.

A different problem with the incremental interpretation of in-situ wh-phrases is the position of the scope marker. As explained in Chapter 3, in-situ wh-phrases have a scope marker that, by use of structural postulates, are displaced towards a position in the sentence, so it can take scope over the desired clause. In a non-incremental interpretation this is not a problem, but for the incremental interpretation algorithm we have described it is a problem. Since the incremental interpretation algorithm calculates intermediate types of the every first two input words (either intermediate or not), when the in-situ wh-phrase is detected, the scope marker has no place to move anymore. To mark scope over a clause the scope marker is always displaced to the left. But when there is only one intermediate type to the left of the scope marker, where should it exactly be displaced to? If it is displaced to the front of the sentence, the scope marker takes scope over the entire question. However, this is not the desired case for all wh-questions. The wh-question *"Who knows who the killer is?"*, for example, requires as an answer a person that knows who the killer is. Should the scope marker for the in-situ wh-phrase *'who'* be displaced to the front of the sentence, an answer given as argument to the semantic term will be substituted for the in-situ gap created by the in-situ wh-phrase. This is certainly undesired behaviour of a scope marker. Should one want to overcome the problem of the incremental interpretation of multiple wh-questions, one should take these two problems into consideration.

However, the presented algorithm is capable of incrementally interpreting direct local wh-questions that only contain one wh-phrase. As that one wh-phrase is wh-fronted, it will always be an ex-situ wh-phrase. It might also be capable of interpreting other variants of wh-questions, such as indirect local wh-questions, or wh-questions with island constraints, but in this thesis we have not tested the algorithm against such questions. A benefit of the presented algorithm is that is easy to implement. The rules of **M♦** are derivation rules, and the programming language Prolog, for instance, lends itself extremely well for an implementation of the derivation rules. Another benefit of the presented algorithm is that it is capable of deriving a semantic term after every input word in a wh-question. We find a semantic term in Continuation-Passing Style, i.e. containing a continuation, for every initial phrase of a question. These initial semantic terms could be used to maximize efficiency of natural language processing in artificially intelligent machines. Combining these two benefits, might make research in the field of incremental interpretation using a typelogical grammar quite relevant to AI.

Chapter 6

Discussion

In this thesis we have established a working algorithm for incremental interpretation, by use of the typelogical grammar extended with system **M♦**. The choice of which words to calculate an intermediate type for with this algorithm is very important for the strength of incrementality. If we restrict the algorithm to always calculate an intermediate type for the first two inputs words (whether the first word is an intermediate type or not), strong incrementality will arise. When we do not restrict the choice of words to calculate an intermediate type for, weak incrementality will arise. Depending on the type of application for which this approach is used, one might want to take on that restriction or not.

This algorithm has been tested on a very small set of sentences, only containing the wh-phrases *'who'*, *'whom'* *'what'*. More research is necessary to adapt the algorithm to other kinds of wh-questions. Other wh-phrases, such as *'when'*, *'why'* and *'how'* are left out of scope, and a full general algorithm for the incremental interpretation of wh-questions, would take those wh-phrases into consideration as well. A benefit of the algorithm that we have presented, however, is that it is easily adaptable. If one wants to interpret different wh-phrases incrementally, it is only necessary to research the context in which those wh-phrases occur, and define wh-type schemata accordingly. As defined by Vermaat [2006], for example, the wh-phrase *'which'* is of type $WH(np, A, wh)/n$, where $A \in \{s, q\}$. It requires a noun on its right hand-side before it is considered to be a wh-phrase. This makes *'which'* an interesting subject for research in incremental interpretation. We do believe, however, that the presented algorithm in this thesis is general enough to interpret such wh-phrases incrementally.

Moreover, the presented algorithm is not capable of interpreting multiple wh-questions incrementally. As stated multiple times, this is due to the fact that the wh-type schema of in-situ wh-phrases decompose to a type that contains a product operator as main connective, and the system **M♦** does not accommodate product operators as main connective with any of its rules. This might also be an interesting subject for further research. Certainly because it is still discussed what a correct answer to a multiple wh-question should be. Would the answer pair *"Everyone someone"* for example be a correct answer to the multiple wh-question *"Who saw whom?"*. In our humble opinion, it very well is a correct answer to that question. However, Vermaat states that an answer to such a question can not contain any generalized quantifiers. According to her, only an answer such as *"Mary John"* would be a correct answer. Considering the correct answer to multiple wh-questions are still debated, research into incremental interpre-

tation of in-situ wh-phrases might achieve very interesting results.

We found a benefit of using a typelogical grammar for this algorithm instead of a CCG in the fact that wh-type schemata had already been defined by Vermaat [2006]. If we had chosen to use a CCG, where no types and semantics for wh-phrases are defined, we had to take that into the scope of the thesis as well. Another benefit of using a typelogical grammar has been found in the fact that the system **M** by Moortgat [1988] is designed in such a way that whenever an intermediate type is found, it has to be derivable by the typelogical grammar. This ensures that when intermediate types can be found throughout the whole process of interpretation, the sentence given as input is also derivable non-incrementally.

The next step of incremental interpretation using a typelogical grammar might take place in a number of ways. First of all, the subject of interpretation may be expanded. In this thesis we have focused on wh-questions, but incremental interpretation may just as well be interesting for any kind of questions, declaratives or even dialogues. Incremental interpretation of other kinds of sentences or dialogues may just as well be modeled the same way we have done in this thesis, i.e. providing an extension to **M**♦. We recommend that anyone who attempts to do this, does a wide literature research of all contexts their subject of incremental interpretation could be in. But not only that could be a point of interest for further research. As we have stated before, wh-phrases are similar to generalized quantifiers and the **M**♦ can also be extended to accommodate them.

Moreover, the subject of interpretation may also be expanded on a broader level. In this thesis, English has been our subject language. It may also be interesting to test our algorithm to other languages and adapt it if necessary. For example, we have very shortly discussed that Dutch is a V2 language, meaning that the main verb is always in the second position of the sentence. Wh-questions such as *"Who saw Bill?"* and *"Whom did Bill see?"* both translate to the Dutch wh-question *"Wie zag Bill?"*. This is a source of ambiguity and might be very interesting to model in an algorithm for incremental interpretation. At a certain point while interpreting, a choice has to be made whether the semantic term is going to correspond to the wh-question questioning the subject or the object.

Something that we have left entirely out of scope in this thesis, is the accounting for answers. We have discussed it in Chapter 3, and after that omitted the interpretation of the answer. However, interpretation of the answer is part of the incremental interpretation of wh-questions, as questions only form a proposition that has a truth value together with their answers. We do think that the algorithm provided in this thesis is capable of interpreting the answer as well. In this case it would simply be application of **M**♦ to the translated types of the question and the answer in that order. This should not result in any problems given the system **M**♦ and the correct semantic term for the answer, i.e. it being of the right type.

The last point of discussion is the value of the semantic terms that are derived for the intermediate types. In the context of an artificially intelligent assistant that is able to answer any kind of question, it might be interesting to enrich the semantic terms with information that can be used regardless of what the eventual term might be. As we have stated before, the strength of incremental interpretation might be seen in artificially intelligent assistants starting to search through the answer space to maximize the efficiency of generating a response. At this point, our

algorithm takes in valuable information to generate the semantic term of the intermediate type, but does not reflect it in the term that is eventually given for the intermediate type. For example, when calculating the semantic term of the phrase *"Who ate..."* it is desirable to incorporate the fact that *'who'* refers to a person or entity into the semantic term of the intermediate type of this phrase. This information is not taken into account yet, as the semantic term we find for intermediate types is simply a semantic term containing what might be called a continuation. Although these intermediate types are a step in the right direction, enriching these terms might be a good starting point for further research. For an actual implementation of incremental interpretation for an artificially intelligent assistant, this might be very valuable together with our algorithm of incremental interpretation.

All in all, we have presented a starting point for modeling incremental interpretation of wh-questions using a typelogical grammar in this thesis. Even though it might not be finished yet, as it is not capable of interpreting multiple wh-questions, it has laid the foundation for incremental interpretation of wh-questions and other types of questions. So, as we posed in the introduction: is it possible to model incremental interpretation using a typelogical grammar? After incrementally interpreting this question, we can now answer this question positively.

Bibliography

G. Altmann and M. Steedman. Interaction with context during human sentence processing. *Cognition*, 30(3):191–238, 1988.

B. R. Ambati, T. Deoskar, M. Johnson, and M. Steedman. An incremental algorithm for transition-based ccg parsing. *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 53–63, 2015.

B. R. Ambati, S. Reddy, and M. Steedman. Assessing relative sentence complexity using an incremental CCG parser. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies*, pages 1051–1057, 2016.

C. Barker and C. C. Shan. Continuations and natural language. *Oxford studies in theoretical linguistics*, 53, 2014.

L. L. S. Cheng. Wh-in-situ. *Glot International*, 7(4):103–109, 2003.

P. De Groote. Towards a Montagovian account of dynamics. *Semantics and Linguistic Theory*, 16:1–16, August 2006.

V. Demberg. Incremental derivations in CCG. *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 11)*, pages 198–206, 2012.

J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2): 94–102, 1970.

U. J. Hopcroft, J.E. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

M. A. Just and P. A. Carpenter. A theory of reading: From eye fixations to comprehension. *Psychological review*, 87(4):329–354, 1980.

A. G. T. Kamide, Y. and S. L. Haywood. The time-course of prediction in incremental sentence processing: Evidence from anticipatory eye movements. *Journal of Memory and language*, 49(1):133–156, 2003.

G. P. Kearsley. Questions and question asking in verbal discourse A cross-disciplinary review. *Journal of psycholinguistic research*, 5(4):355–375, 1976.

D. E. Knuth. On the translation of languages from left to right. *Information and control*, 8(6): 607–639, 1965.

G. R. Kuperberg, T. Sitnikova, D. Caplan, and P. J. Holcomb. Electrophysiological distinctions in processing conceptual relationships within simple sentences. *Cognitive brain research*, 17(1): 117–129, 2003.

N. Kurtonina and M. Moortgat. Structural control. *Specifying syntactic structures*, pages 75–113, 1997.

J. Lambek. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3): 154–170, 1958.

E. Lebedeva. *Expressing discourse dynamics through continuations.* PhD thesis, Université de Lorraine, 2012.

V. Lombardo and P. Sturt. Incremental processing and infinite local ambiguity. *Proc. 19th Annual Conference of the Cognitive Science Society*, pages 448–453, August 1997.

V. Lombardo and P. Sturt. Incrementality and lexicalism: A treebank study. *The lexical basis of sentence processing formal, computational and experimental issues*, pages 137–155, 2002.

W. Marslen-Wilson. Linguistic structure and speech shadowing at very short latencies. *Nature*, 244(5417):522–523, 1973.

D. Milward and R. Cooper. Incremental interpretation: Applications, theory, and relationship to dynamic semantics. *Proceedings of the 15th conference on Computational linguistics*, 2: 748–754, August 1994.

M. Moortgat. Categorial investigations: Logical and linguistics aspects of the lambek calculus. *Groningen-Amsterdam Studies of Semantics*, 1988.

M. Moortgat. Generalized quantifiers and discontinuous type constructors. *Natural Language Processing*, 6:181–208, 1996.

M. Moortgat. Categorial type logics. *Handbook of logic and language*, pages 93–177, 1997.

M. Moortgat. Constants of grammatical reasoning. *Constraints and resources in natural language syntax and semantics*, pages 195–219, 1999.

A. Munn. Island constraints, 2016. Retrieved on August, 2016 from https://www.msu.edu/course/lin/434/PSets/island-constraints.pdf.

J. C. Reynolds. The discoveries of continuations. *Lisp and symbolic computation*, 6(3-4):233–247, 1993.

C. Rudin. On multiple questions and multiple wh fronting. *Natural Language and Linguistic Theory*, 6(4):445–501, 1988.

R. E. Stearns and P. M. Lewis. Property grammars and table machines. *Information and Control*, 14(6):524–549, 1969.

M. Steedman. *The syntactic process*, volume 24 of *Language, Speech and Communication.* MA: MIT press, 2000.

J. van Benthem. The semantics of variety in categorial grammar. *Categorial grammar*, 25:37–55, 1988.

W. K. Vermaat. *The Logic of Variation: A cross-linguistic account of wh-question formation.* PhD thesis, Utrecht University, 2006. Utrecht Institute of Linguistics OTS.

D. H. Younger. Recognition and parsing of context-free languages in time n3. *Information and control*, 10(2):189–208, 1967.

Y. Zhang and S. Clark. Shift-reduce CCG parsing. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics Human Language Technologies*, 1:683–692, June 2011.

Appendix A

Rules of the typelogical grammar

A.1 Without semantic terms

$$\frac{}{A \vdash A}\ Ax$$

$$\frac{\Delta \circ B \vdash A}{\Delta \vdash A/B}\ [/I]$$

$$\frac{B \circ \Delta \vdash A}{\Delta \vdash B\backslash A}\ [\backslash I]$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma \circ \Delta \vdash A \bullet B}\ [\bullet I]$$

$$\frac{\blacklozenge(\Gamma) \vdash A}{\Gamma \vdash \Box A}\ [\Box I]$$

$$\frac{\Gamma \vdash A}{\blacklozenge(\Gamma) \vdash \Diamond A}\ [\Diamond I]$$

$$\frac{\Gamma \vdash A \quad A, \Delta \vdash B}{\Gamma, \Delta \vdash B}\ [Cut]$$

$$\frac{\Gamma \vdash A/B \quad \Delta \vdash B}{\Gamma \circ \Delta \vdash A}\ [/E]$$

$$\frac{\Delta \vdash B \quad \Gamma \vdash B\backslash A}{\Delta \circ \Gamma \vdash A}\ [\backslash E]$$

$$\frac{\Delta \vdash A \bullet B \quad \Gamma[(A \circ B)] \vdash C}{\Gamma[\Delta] \vdash C}\ [\bullet E]$$

$$\frac{\Gamma \vdash \Box A}{\blacklozenge(\Gamma) \vdash A}\ [\Box E]$$

$$\frac{\Delta \vdash \Diamond A \quad \Gamma[\blacklozenge(A)] \vdash B}{\Gamma[\Delta] \vdash B}\ [\Diamond E]$$

A.2 With semantic terms

$$\frac{}{x : A \vdash x : A}\ Ax$$

$$\frac{\Delta \circ x : B \vdash u : A}{\Delta \vdash \lambda x.u : A/B}\ [/I] \qquad \frac{\Delta \vdash u : A/B \quad \Gamma \vdash v : B}{\Delta \circ \Gamma \vdash (u\ v) : A}\ [/E]$$

$$\frac{x : B \circ \Delta \vdash u : A}{\Delta \vdash \lambda x.u : B\backslash A}\ [\backslash I] \qquad \frac{\Delta \vdash v : B \quad \Gamma \vdash u : B\backslash A}{\Delta \circ \Gamma \vdash (u\ v) : A}\ [\backslash E]$$

$$\frac{\Delta \vdash t : A \quad \Gamma \vdash u : B}{\Delta \circ \Gamma \vdash \langle t, u \rangle : A \bullet B}\ [\bullet I] \qquad \frac{\Delta \vdash u : A \bullet B \quad \Gamma[x : A \circ y : B] \vdash t : C}{\Gamma[\Delta] \vdash t[\pi^1 u/x, \pi^2 u/y] : C}\ [\bullet E]$$

A.3 Postulates

Associativity

$$\frac{\Gamma[(\Delta_1 \circ \Delta_2) \circ \Delta_3] \vdash C}{\Gamma[\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash C}\ [Ass]$$

Left displacement postulates

$$\frac{\Gamma[(\blacklozenge\Delta_1 \circ \Delta_2) \circ \Delta_3] \vdash C}{\Gamma[\blacklozenge\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash C}\ [Pl1] \qquad \frac{\Gamma[\Delta_2 \circ (\blacklozenge\Delta_1 \circ \Delta_3)] \vdash C}{\Gamma[\blacklozenge\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash C}\ [Pl2]$$

Right displacement postulates

$$\frac{\Gamma[\Delta_1 \circ (\Delta_2 \circ \blacklozenge\Delta_3)] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2) \circ \blacklozenge\Delta_3] \vdash C}\ [Pr1] \qquad \frac{\Gamma[(\Delta_1 \circ \blacklozenge\Delta_3) \circ \Delta_2] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2) \circ \blacklozenge\Delta_3] \vdash C}\ [Pr2]$$

Controlled associativity

$$\frac{\Gamma[\blacklozenge(\Delta_1 \circ \blacklozenge(\Delta_2 \circ \Delta_3))] \vdash C}{\Gamma[\blacklozenge(\blacklozenge(\Delta_1 \circ \Delta_2) \circ \Delta_3)] \vdash C}\ [Ass_\blacklozenge]$$

Controlled reordering/displacement

$$\frac{\blacklozenge(\blacklozenge(\Delta_1 \circ \blacklozenge\Delta_3) \circ \Delta_2) \vdash \Gamma}{\blacklozenge(\blacklozenge(\Delta_1 \circ \Delta_2) \circ \blacklozenge\Delta_3) \vdash \Gamma}\ [Pr2_\blacklozenge]$$

Appendix B

Rules of M and M♦

B.1 Without semantic terms

M:

$$
\begin{array}{rlll}
R1_l: & X/Y, & Y & \Rightarrow X \\
R1_r: & Y, & Y\backslash X & \Rightarrow X \\
\\
M1_l: & Y\backslash X, & Z & \Rightarrow Y\backslash W \quad \text{where} \\
 & X, & Z & \Rightarrow W \\
M1_r: & Z, & X/Y & \Rightarrow W/Y \quad \text{where} \\
 & Z, & X & \Rightarrow W \\
\\
M2_l: & X/Y, & Z & \Rightarrow X/W \quad \text{where} \\
 & Z, & W & \Rightarrow Y \\
M2_r: & Z, & Y\backslash X & \Rightarrow W\backslash X \quad \text{where} \\
 & W, & Z & \Rightarrow Y
\end{array}
$$

M♦ :

$$
\begin{array}{rlll}
R1_l\text{:} & \blacklozenge(\Box X/Y & \circ\ Y) & \Rightarrow X \\
R1_r\text{:} & \blacklozenge(Y & \circ\ Y\backslash\Box X) & \Rightarrow X \\
 & & & \\
R2\text{:} & \blacklozenge(\Box X/(\Box Y1/\blacklozenge\Box Y2) & \circ\ \Box Y1/Y2) & \Rightarrow X \\
 & & & \\
M1_r\text{:} & \blacklozenge(Z & \circ\ \Box X/Y) & \Rightarrow \Box W/Y \quad \text{where} \\
 & \blacklozenge(Z & \circ\ X) & \Rightarrow W \\
 & & & \\
M2_l\text{:} & \blacklozenge(\Box X/Y & \circ\ Z) & \Rightarrow \Box X/W \quad \text{where} \\
 & \blacklozenge(Z & \circ\ W) & \Rightarrow Y \\
 & & & \\
M2\blacklozenge\text{:} & \blacklozenge(\Box X/(\Box Y1/\blacklozenge\Box Y2) & \circ\ Z) & \Rightarrow \Box X/W \quad \text{where} \\
 & \blacklozenge(Z & \circ\ Y2) & \Rightarrow \Box Y1/W
\end{array}
$$

B.2 With semantic terms

M :

$$
\begin{array}{rlll}
R1_l: & \mathrm{X/Y}: \mathsf{Functor}, & \mathrm{Y}: \mathsf{Arg} & \Rightarrow \mathrm{X}: \mathsf{Functor\ Arg} \\
R1_r: & \mathrm{Y}: \mathsf{Arg}, & \mathrm{Y\backslash X}: \mathsf{Functor} & \Rightarrow \mathrm{X}: \mathsf{Functor\ Arg} \\
 & & & \\
M1_l: & \mathrm{Y\backslash X}: \mathsf{Functor}, & \mathrm{Z}: \mathsf{SemZ} & \Rightarrow \mathrm{Y\backslash W}: \lambda\mathsf{VarY.\ SemW} \quad \text{where} \\
 & \quad \mathrm{X}: \mathsf{Functor\ VarY}, & \mathrm{Z}: \mathsf{SemZ} & \Rightarrow \mathrm{W}: \mathsf{SemW} \\
M1_r: & \mathrm{Z}: \mathsf{SemZ}, & \mathrm{X/Y}: \mathsf{Functor} & \Rightarrow \mathrm{W/Y}: \lambda\mathsf{VarY.\ SemW} \quad \text{where} \\
 & \quad \mathrm{Z}: \mathsf{SemZ}, & \mathrm{X}: \mathsf{Functor\ VarY} & \Rightarrow \mathrm{W}: \mathsf{SemW} \\
 & & & \\
M2_l: & \mathrm{X/Y}: \mathsf{Functor}, & \mathrm{Z}: \mathsf{SemZ} & \Rightarrow \mathrm{X/W}: \lambda\mathsf{VarW.\ (Functor\ SemY)} \quad \text{where} \\
 & \quad \mathrm{Z}: \mathsf{SemZ}, & \mathrm{W}: \mathsf{VarW} & \Rightarrow \mathrm{Y}: \mathsf{SemY} \\
M2_r: & Z: \mathsf{SemZ}, & Y\backslash X: \mathsf{Functor} & \Rightarrow W\backslash X: \lambda\mathsf{VarW.\ (Functor\ SemY)} \quad \text{where} \\
 & \quad \mathrm{W}: \mathsf{VarW}, & \mathrm{Z}: \mathsf{SemZ} & \Rightarrow \mathrm{Y}: \mathsf{SemY}
\end{array}
$$

M♦ :

$R1_l$: $\blacklozenge(\Box X/Y$: Functor $\circ\ Y$: Arg) $\Rightarrow X$: Functor Arg

$R1_r$: $\blacklozenge(Y$: Arg $\circ\ Y\backslash\Box X$: Functor) $\Rightarrow X$: Functor Arg

$R2$: $\blacklozenge(\Box X/(\Box Y1/\blacklozenge\Box Y2)$: Functor $\circ\ \Box Y1/Y2$: Arg) $\Rightarrow X$: Functor Arg

$M1_r$: $\blacklozenge(Z$: SemZ $\circ\ \Box X/Y$: Functor) $\Rightarrow \Box W/Y$: λVarY. SemW where
$\blacklozenge(Z$: SemZ $\circ\ X$: Functor VarY) $\Rightarrow W$: SemW

$M2_l$: $\blacklozenge(\Box X/Y$: Functor $\circ\ Z$: SemZ) $\Rightarrow \Box X/W$: λVarW. (Functor SemY) where
$\blacklozenge(Z$: SemZ $\circ\ W$: VarW) $\Rightarrow Y$: SemY

$M2\blacklozenge$: $\blacklozenge(\Box X/(\Box Y1/\blacklozenge\Box Y2)$: Functor $\circ\ Z$: SemZ) $\Rightarrow \Box X/W$: λVarW. (Functor (λVarY2. SemY1)) where
$\blacklozenge(Z$: SemZ $\circ\ Y2$: VarY2) $\Rightarrow \Box Y1/W$: λVarW. SemY1