2 Theoretical backgrounds

In this chapter I will briefly sketch the main theoretical assumptions and frameworks that underlie the analyses of single and multiple negation in this book. This chapter will cover three different fields in linguistic theory: minimalist syntax, truth-conditional semantics and the syntax-semantics interface. The first section will cover the minimalist program; section 2.2 will provide an overview of truth-conditional semantics and those approaches of semantic theory that are relevant for this study, and section 2.3 will capture the main notions belonging to the syntax-semantics interface. The main purpose of this chapter is to describe the theoretical tools that I need for my analysis of negation. Those include both a syntactic apparatus and semantic machinery in order to describe and explain correct grammatical structures and the correct interpretations. The first two sections will have a descriptive nature: I will restrict myself to describing the syntactic and semantic notions that are necessary for the analyses. It will turn out that the domains of syntactic and semantic theories overlap with respect to certain phenomena, and I discuss where syntax and semantics meet. As a result of this overlap, the third section does not only consist of a descriptive discussion of current theories, but also contains critical comparison between the syntactic and semantic strategies.

2.1 The Minimalist Program

In this section I will provide a brief overview of the minimalist syntactic theory that has been developed in the last decade. In subsection 2.1.1 I will sketch the model of grammar that forms the basis of the Minimalist Program. In subsection 2.1.2 I will discuss the notions of interpretable and uninterpretable features and the mechanism of feature checking. Subsection 2.1.3 will cover the three basic syntactic operations: Merge, Move and Agree. Finally, in subsection 2.1.4 I will explain Chomsky's (2001) ideas on locality and phase theory.

2.1.1 A minimalist model of grammar

The Minimalist Program (Chomsky 1992, 1995, 1998, 2000, 2001) is an elaboration of the Principle and Parameters framework, designed such that it requires only a minimum of theoretical apparatus. The Principle and Parameters framework states that the human language faculty consists of a set of universal principles and a set of parameters, which constitute linguistic variation. In the Minimalist Program, a research program guided by Ockam's methodological principles, rather than a fixed linguistic theory, language is thought of as a (nearly) optimal linking between linguistic form and linguistic meaning. Linguistic expressions are generated in the linguistic component (the Language Faculty $F_L$) of the mind that has interfaces with the articulatory-perceptual (AP)
system and LF the Conceptual-Intentional (CI) system. Form and meaning are represented at these two interfaces, which are the only levels of representation within the theory. PF (phonological form) is the interface between FL and the AP system and LF (Logical Form) is the interface between FL and the CI system.

(1) The linguistic component and its interfaces with other components

Each linguistic expression can be seen as a tuple \(<\pi, \lambda>\), where \(\pi\) stands for the phonological form (the sound or gestures) of a linguistic expression and \(\lambda\) for the logical form of the expression (the meaning). A linguistic expression is a single syntactic object that is derived during the computation of the expression.

The lexicon consists of lexical entries, each containing lexical items (LI’s). LI’s are thought of as bundles of features. Chomsky (1995) distinguishes three different kinds of features: phonological features, semantic features and formal features. The set of phonological features of a certain LI encodes all the information that is needed at PF to enter the AP system. An example of a phonological feature is ‘ending on a /d/’ (\(\_\_\_\_d/\#\)). Semantic features are those features that can be interpreted at LF, e.g. [+animate]. Formal features are categorial features like [+V] or so-called φ features on verbs that contain the information about number, gender or case. These features encode information for the syntactic component. Formal features are either interpretable [iFF] or uninterpretable [uFF]. Interpretable means legible at LF, i.e. containing semantic contents; uninterpretable features cannot be interpreted at LF or PF. The fact that these features are not legible at the interfaces violates the principle of Full Interpretation (Chomsky 1995) that says that the syntactic objects at the interfaces should be fully interpretable and therefore may not contain any uninterpretable features. Hence uninterpretable features need to be deleted during the derivation, to prevent the sentence from crashing at the interfaces (see 2.1.2 on feature checking and 2.1.3 on Agree).

A consequence of the assertion that PF and LF are the only available levels of representation is that syntactic principles can only apply on these levels. Due to the fact that the only two levels of linguistic representation are interfaces, principles can only operate on the interface between syntax and phonology or syntax and semantics. There is no pure syntactic level as previous theories of grammar postulated. This
reduces the number of purely syntactic principles (in the ideal situation) to 0. As a result of this, parameters can no longer be regarded as part of the core of (pure) syntax. Because the interfaces interact with other components (namely the AP and CI system), conditions on the interface cannot be thought of as subject to cross-linguistic (i.e. parametric) variation. Without any other level of representation, the only remaining locus for parametric variation is the lexicon. Therefore cross-linguistic variation (as well as language-internal variation) is the result of lexical variation: the formal properties of lexical items encode all necessary information for the syntactic derivation, and differences with respect to these formal properties lead to linguistic variation.

\[\text{(2) Model of Grammar (Chomsky 1995)}\]

The figure in (2) shows that a set of lexical items enter a numeration \(N\), which is a set of pairs \(<LI, i>\), whereby LI is a lexical item and \(i\) the number of its occurrences in \(N\). Every time a lexical item from \(N\) enters the derivation of the expression \(<\pi, \lambda>\), \(i\) is reduced by 1 until every index of every lexical item is 0. If not, the derivation crashes. The derivation can be seen as a mapping from \(N\) onto the set of linguistic expressions \(<\pi, \lambda>\).

At a certain point during the derivation, the phonological features are separated from the formal and semantic features. This moment is called Spell-Out. At this stage of the derivation, the phonological features are mapped onto PF, whereas the formal and semantic features follow their way towards LF. After Spell-Out, syntactic operations still take place, both between Spell-Out and LF and between Spell-Out and PF. However, operations between Spell-Out and PF do not influence LF, and operations between Spell-Out and LF do not influence PF.
2.1.2 Feature checking and functional projections

As has been shown in the previous subsection, the role of the derivation is to create syntactic objects (sentences) that do not crash at the interfaces. Therefore the derivation needs to delete all uninterpretable features. Uninterpretable formal features can be deleted by means of feature checking, a mechanism that allows a category to check its uninterpretable feature against an interpretable feature of the same category. Hence categories sharing the same formal features establish syntactic relationships. Chomsky (1995) argues that feature checking takes place in specifier-head configuration: interpretable features in a spec position can check uninterpretable features in a head position and vice versa.

Hence the distinction between heads (syntactic elements that project themselves) and specifiers (modifiers of the head that remain under the projection of the head) is crucial for these relationships. Checking theory requires that in every checking relation both a syntactic head and a specifier are involved. As a consequence the number of syntactic categories should be expanded by a number of functional categories in which feature checking can take place. The fact that for every checking relation a syntactic head is involved links the availability of any uninterpretable feature \([uF]\) to the availability of a syntactic category \(F\). For every uninterpretable feature \([uF]\) there is a functional category \(F\) and checking takes place in \(FP\) under spec head configuration.

An example of a functional projection is DP, in which the head is occupied by an uninterpretable \([uDET]\) feature that establishes a checking relation with an element carrying an interpretable \([iDET]\) in Spec,DP. Thus the deletion of the uninterpretable \([uDET]\) feature in \(D^o\) can take place. Before discussing the mechanism of feature checking in detail, I will first discuss the possible syntactic operations during the derivation.

2.1.3 Syntactic operations

Three different syntactic operations play a role in syntax: Merge, Agree and Move. Since Move can be described in terms of Merge, this leaves two independent operations left to establish syntactic relationships.

Merge is the operation that takes two elements from the numeration \(N\) and turns them into one constituent that carries the same label as that of the dominating item. The notion of labeling replaces the previous notion of X-Bar structure.

---

5 Epstein (1995) and Zwart 2004 argue that the relation Spec-head does not reflect a mathematical relation in the structure and therefore propose to replace spec-head agreement by the notion of sisterhood (which contrary to the spec-head configuration can be captured in mathematical terms).

6 Note that this does not a priori exclude the availability of the category \(F\) if there is no uninterpretable feature \([F]\). Morphological words carrying an interpretable \([F]\) feature could be base-generated in an \(F^o\) position too.
The difference between labeling after Merge and X-Bar structure is that X Bar structure requires a tripartite structure that consists of a head, a complement and a (possibly empty) specifier. Merge generates bipartite structures that either consist of a head and a complement or of a specifier/adjunct and a head. Labeling theory only assigns the label (i.e. the syntactic category) of the dominant category \((a/b)\) to the complex \(\{a, \beta\}\).\(^7\) The combination of a transitive verb \(V\) and an object \(D\) for example yields an (intransitive) verb, so the label of the terminal node \(V\) is also the label of the branching node \(\{V, D\}\). Note that these structures, referred to by Chomsky (1995) as Bare Phrase Structures (as opposed to X-Bar structures), lack prefabricated maximal projections. The notion of maximal projection is reduced to the highest instance of a syntactic category \(X\). Note that the only operation that generates structure is Merge, so if there is no specifier available, the merge of a complement \(\beta\) with a head \(\alpha\) will be the highest instance of \(\alpha\), and \(\alpha\) can be merged with a new head. The operation Merge is defined as in (3):

\[(3) \quad \text{Merge: } K = \{a\beta \{a, \beta\}\}\]

\(K\) is a newly-formed constituent that is labeled after its head which can be either \(\alpha\) or \(\beta\). Two options are available. Either \(K\) merges with a new head \(\gamma\) yielding a new constituent \(L_1\) labeled \(\gamma\) ((4)a) or it is merged with an LI \(\delta\) that is not a head, yielding \(L_2\) with label \(\alpha\) where \(\delta\) is called a specifier ((4)b). This latter constituent can be merged in its turn with a head (yielding \(L_3\)) ((4)c), similar to the case of ((4)a) or with a non-head \(\epsilon\) ((4)c). In the latter case there is more than one specifier (\(\delta\) and \(\epsilon\)), and \(\epsilon\) is called an adjunct of \(\alpha\).

\[(4)\]
\[
\begin{align*}
\text{a. } & L_1 = [\gamma \ y \ [a \ \alpha \ \beta]]^8 \\
& [p \ \text{in \ } [\text{o \ the \ sky}]] \\
\text{b. } & L_2 = [a \ \delta \ [a \ \alpha \ \beta]] \\
& [p \ \text{high \ } [p \ \text{in \ } [\text{the \ sky}]]] \\
\text{c. } & L_3 = [a \ \epsilon \ [a \ \delta \ [a \ \alpha \ \beta]]] \\
& [p \ \text{still \ } [p \ \text{high \ } [p \ \text{in \ } [\text{the \ sky}]]]]
\end{align*}
\]

The second operation that may follow is Move. Move is an operation that is derived from Merge. Instead of merging two constituents from the numeration \(N\), it is also possible to merge \(K\) with a subpart of \(K\).

\[(5) \quad \text{Move: } L = \{a, K\}, \text{ whereby } K = \{\gamma \ \{\ldots \ a \ldots \ \}\} \]

Move is an operation that takes a few steps. Suppose that \(\alpha\) is a term of some constituent \(K\) and for whatever reason \(\alpha\) has to raise to a position to the left of \(K\). In that case \(\alpha\) will be copied into two identical constituents, and \(K\) merges with the copy of \(\alpha\) yielding \(L = \{\gamma \ [a, K(=\{\ldots \ a \ldots \})]\}\). After this movement the second \(\alpha\) is

---

\(^7\) See Collins (2002) for a framework without any labelling.

\(^8\) The choice for \(\alpha\) as the label of the Merge of \(\alpha\) and \(\beta\) is arbitrary. It could also be \(\beta\), as long as every higher label of \(\alpha\) is replaced by \(\beta\) too.
phonologically no longer visible. For the rest, the structure remains unaffected (Chomsky 1995: 250). This means that the so-called trace of α does not get deleted, but will only be marked for its phonological invisibility. This mechanism is commonly referred to as the copy theory of movement. One of its most famous applications is Wh-movement in which a Wh-element is fronted from its base-generated position.

(6) \[\text{what } [\text{did you eat what}]\]

In (6) there are two instances of what, but only one will be spelled out. The rule that specifies that the first Wh element will be spelled out in sentence-initial position (at least in English) is a condition on PF.

The third syntactic operation is Agree\(^9\). Agree is the operation that establishes a relation between two features of the same kind. If an LI consists of a feature that is uninterpretable, it needs to check this feature against a feature of the same kind. It is said that the category probes for a goal.

In the older versions of minimalism (Chomsky 1995) uninterpretable features were deleted after checking against an interpretable feature. However, the notion of interpretation is defined as a semantic or phonological notion: a feature that has semantic or phonological content is interpretable. Therefore syntax had to ‘look ahead’ for the semantics or phonology in order to allow feature checking or not, but a derivation is not supposed to have any ‘contact’ with the interface before reaching them.\(^{10}\)

In order to solve this problem, Chomsky (1999) proposes a new system in terms of valuation. Some features, like [Def] or [Case], in the lexicon do not have any value (definite/indefinite, or nominative/accusative, etc.) yet. During the derivation these features can be valued by means of Agree. All (unvalued) features need to be valued: a derivation containing unvalued features will crash at the interfaces.

Valuation takes place by Agree when a properly valued (interpretable) feature is in specifier head (spec head) relation with an unvalued feature. A good example is subject verb agreement: the finite verb has φ features (such as person and number) that are semantically vacuous on the verb. The subject has the same kind of features, but these are meaningful for pronouns or DP’s. The subject (being a DP) also has an unvalued [Case] feature that will be valued [Nom] by the Agree relation with the finite verb. So, Agree is a two-way relation between two lexical items that valuate

\(^9\) For the texts on Agree, I thankfully made use of Kremers’ (2003) explanation of this operation.

\(^{10}\) Note that this forms a huge problem for the syntactic model: if the numeration is not allowed to be in a transparent relation with LF throughout the derivation, the question rises what triggers the numeration N in the first place. The set of selected elements should correspond to the meaning of the sentence, which is represented at LF. Chomsky (p.c.) argues that the numeration is open to ambiguity, e.g. “the cat bites the dog” is derived from the (2*), cat, bites and dog, which could also yield the sentence “the dog bites the cat.” Even if this were possible (which is doubtful as the lexical elements are said to enter the derivation with all the case features with them), it remains unclear why these lexical items have been triggered and others not. Hence, there should be some relation between the meaning of the sentence (at least the intended meaning) and LF.
(some of) each other's unvalued features. At LF, valued features without semantic content on the LI can be deleted.

\[(7) \quad [V_{\text{fin}} [v \quad SU]]
[\text{Case}_{\text{unv}}] [\text{Nom}_{\text{val}}]
[\varphi_{\text{val}}] \uparrow [\varphi_{\text{unv}}] \]

In (7) the subject enters the derivation with an unvalued [Case] feature and valued $\varphi$ features whereas the finite verb enters the derivation with unvalued $\varphi$ features but with a valued case-feature. Therefore Agree values both sets of unvalued features. As $\varphi$ features are meaningless on verbs and Case has no meaning at all, all features except for the subject's $\varphi$ features will be deleted at LF\(^{11}\).

Note that this Agree relation is a relation on distance. This is not always the case. In some cases (in fact in many languages, this happens to be the case for verb-subject agreement), it is required that the subject moves to a specifier position of T. In a Bare Phrase model of grammar, this is a problem since heads do not necessarily require a (possibly empty) specifier. To solve this problem, Chomsky (1999, 2001) assumes that in these languages the [Tense] feature is accompanied by a sub-feature of [Tense], namely the [EPP] feature. The [EPP] feature generates a specifier position (Spec,TP\(^{12}\)) to which the subject may move. Then Agree can take place within the maximal projection of T.

\[(8) \quad \text{Agree after subject movement to } T\]

\[\text{Move} \]

\[\text{[T \quad SU]} \quad \text{[T[EPP]} V_{\text{fin}} [v_{\text{P}} \quad SU]]\]
[\text{Nom}_{\text{unv}}] [\text{Case}_{\text{val}}]
[\varphi_{\text{val}}] \uparrow [\varphi_{\text{unv}}] \]

Now, the issue of locality has been left as a problem reflecting two questions. The first question concerns all three syntactic operations, namely how the maximal distance between two constituents is defined such that a syntactic operation, like

---

\(^{11}\) The minimalist program tries to exclude features that have no semantic content at all. This leads to a puzzle for Case as all Case features will be deleted at LF. Pesetzký and Torrego (2001) therefore argue that nominative case is in fact an (uninterpretable) tense feature. In that case Agree evaluates the unvalued $\varphi$ features on the verb (to be deleted at LF) and the unvalued [Tense] feature on the subject (also to be deleted at LF).

\(^{12}\) Although TP is an inappropriate notion within Bare Phrase structures, it is still commonly used to express the traditional maximal projection (in this case the highest instance of T).
Move is possible. This question will be answered in the next subsection. The second question only addresses the issue of Agree: what will happen if there is more than one active goal in the correct checking domain?

Chomsky (1999) argues that Agree relation between $\alpha$ and $\beta$ can only be realised if $\alpha$ and $\beta$ match (in terms of features) and there is no intervening $\gamma$ such that $\gamma$ also matches with $\alpha$. Chomsky calls this the defective intervention effect\(^{13}\), formalised as a filter as in (9).

\[ *\alpha > \beta > \gamma, \text{ whereby } \alpha, \beta, \gamma \text{ match and } > \text{ is a c-command relation.} \]

However, this constraint is too strong. Ura (1996) and Haraiwa (2001) show there are phenomena in which one head licenses more than one constituent, like Japanese licensing of multiple nominatives by a single $\nu^c$ head in raising constructions (10)\(^{14}\). In this example the multiple instances of nominative case stand in an Agree relation with the single finite verb *kanjita* ‘thought’. Therefore Haraiwa proposes a reformulation of Agree that says that within the proper checking domain $\alpha$ can license both $\beta$ and $\gamma$, unless $\beta$ matches with $\gamma$ and has already been valued by a probe other than $\alpha$ in an earlier stage of the derivation. In other words, Multiple Agree takes place at a simultaneous point in the derivation. I will adopt this theory of Agree for my analysis of Negative Concord.

(10) Johnga [yosouijouni nihonjinga eigoga hidoku] kanjita.\(^{15}\) Japanese

John.NOM than-expecte d the.Japanese.NOM English.NOM bad.INF think.PAST

‘It seemed to John that the Japanese are worse at speaking English than he had expected’

### 2.1.4 Multiple Spell-Out

In the picture of minimalist syntax that I have sketched so far, the domain in which syntactic relations can take place seems to be restricted to the entire derivation. In principle Agree or Move relations can be established between every two (or more) elements in the derivation. It is impossible to assume filters or any other bans on certain kinds of relations during the derivation as the only loci of determination of the grammaticality of a sentence are LF or PF. Those are the only locations where the derivation may converge or not. However, this would make all kinds of movements or Agree relationships possible, which are ruled out in natural language. Therefore it is assumed Spell-Out occurs more than once during the derivation. This means that some parts of the derivation are spelled out and move to LF and PF whereas the

---

\(^{13}\) Note that this is basically a reformulation of Rizzi’s (1989) Relativised Minimality.

\(^{14}\) Other examples are Icelandic licensing of multiple accusatives, or overt multiple Wh fronting in Slavic languages.

\(^{15}\) From Hiraiwa (2001). Japanese is a language in which infinitives fail to assign (nominative) case. All case markings in this sentence have thus been licensed by the matrix light verb.
derivation of the rest of the sentence continues. After Spell-Out these parts are no longer accessible to the rest of the derivation thereby preventing agree-effects over such long distances.

Chomsky refers to these units as phases. The content of a phase is not accessible to the rest of the syntactic system except for the outer layer, the so-called phase edge, consisting of the highest head and its specifier(s). This means that an element, in order to move to a position outside the phase, first has to move to one of the phase edges before it can move to the final destination. Agreement can only take place within the same phase or between an element in the phase plus the edge of the phase that is immediately dominated. Note that what is inside the phase is no longer accessible to syntactic operations, rather the phase as a whole is. Phases can be fronted or extraposed, etc.

According to Chomsky, the fact that a phase is no longer open to syntactic operations is derived from its propositional nature. In a framework in which the subject is base-generated in Spec, vP, the smallest propositional constituent is vP. The other candidate is the projection that roofs the clause, namely CP. VP cannot be a candidate since it lacks a subject, and TP is not a candidate because essential elements of the clause like focus or topic markers are outside TP.

Idiomatic expressions are good examples of elements that have to be interpreted in one and the same phase. In that case they can have their idiomatic reading (11)a. Parts of idiomatic expressions may escape from the vP through a landing-site in Spec,vP if a copy is still in vP in order to be interpreted (11)b. In (11)c the adverb niet forms a negative island from which manner adverbs cannot escape (Honcoop 1998). The first part of the idiomatic expression ‘met zijn neus’ cannot be base-generated inside the phase vP, and hence the idiomatic reading becomes unavailable.

16 Note that the notion of phase stems from the notion of Cycle (Chomsky 1973, 1977). Cycles can be thought of as domains (clauses) for syntactic operations. Once an operation has taken place that moves an element out of cycle 1 into a higher cycle 2, no other syntactic operation is allowed to apply in the cycle 1.
Currently, it is also assumed that DP’s form a phase (Matushansky 2002) or that some kinds of VP’s are phases (Legate 1998). Abels (2003) suggests that PP’s also exhibit phase-like behaviour. Epstein and Seely (2002b) even propose that every maximal projection is a phase. However, the question which projections are phases and which not is beyond the scope of this study. For the analysis of negation, the most important fact is that v\textsuperscript{18} and C are heads of a phase edge.\textsuperscript{19}

Note that the theory of phases leaves us with a problem for agreement over the phase boundary. Still we can find many examples in natural language agreement relations between elements in different phases (without previous stages of the derivation where the two elements are in the same phases). Wh-in-situ is a good example but so is subject-verb agreement in SOV languages: phase theory would block the following Agree relation in Dutch subordinate clauses since the verb is located in a position lower than v\textsuperscript{0} and T is in a position higher than vP.

(12) \ldots dat [\textsubscript{T} Marie [\textsuperscript{v} vaak [\textsubscript{v} slaapt]]]

\ldots that Mary often sleeps

Spell-Out fails to be the distinction between overt and covert movement since V\textsubscript{fin} cannot escape out of the phase through LF movement after Spell-Out (as a result of the cyclic nature spell-out). This requires a more fundamental approach of covert movement. Apparently the finite verb in (12) did move under the assumptions of phase theory: otherwise the Agree relation could not have been established. But the phonological content, i.e. the phonological features, did not move along with the

\textsuperscript{17} This sentence is not ungrammatical. It maintains its literal reading. The idiomatic reading however is no longer available, hence the asterisk in front of the 3\textsuperscript{rd} line.

\textsuperscript{18} This only holds under the assumption that all verbal phrases have a v\textsuperscript{0} head. Originally v\textsuperscript{0} has been introduced to distinguish ergative from unergative VP's. Only unergative verbs can be headed by v\textsuperscript{0}. Under these analyses the v phase should be reformulated as the phases headed by the highest V head in a Larsonian VP shell (Larson 1988).

\textsuperscript{19} Locality forms the core of syntactic theory, and it should be acknowledged that the theory sketched here is far from complete. For example, not all locality effects can be reduced to phases (e.g. relativized minimality (Rizzi 1989)). Phase theory also faces several problems. However, in this chapter I will leave these facts outside the discussion. Locality effects that are of importance for the theory of negation I present in this book will be introduced in due course.
formal (and semantic features) of the finite verb. Hence covert movement should be understood as feature movement from one head position to another head position (in this case \( v \) to \( T \) movement).

This leads us to at least two kinds of movement: \textit{overt movement} (in which a LI or constituent raises to a higher position, leaving a non-spelled out copy) and \textit{feature movement} (in which the formal (and semantic) features move along, adjoining to a higher head).

\begin{equation}
\begin{array}{c}
X \\
Y \\
X \\
\end{array}
\end{equation}

\begin{equation}
\begin{array}{c}
[FF] \\
X \\
V \\
Z \\
\end{array}
\end{equation}

To summarise, the minimalist program tries to reduce syntactic principles to interface conditions, leaving only two operations in the core of the linguistic component: Merge and Agree and conditions on locality. Given that, ideally, Agree only establishes relations between interpretable and uninterpretable elements, even Agree can be thought of as an interface operation. Thus syntactic theory can even be reduced to the operation Merge and locality restrictions, e.g. in terms of phases. Within this framework I will formulate my analysis for the interpretation of (multiple) negation.

\section*{2.2 Truth-conditional semantics}

In this section I will discuss some important notions in semantic theory that I will use for the analysis of negation. In the first subsection I will briefly describe how the current semantic theories that I will use are built on Frege's \textit{Principle of Compositionality}. After that I will outline Lambda Calculus and type theory and show how it can be used to formulate compositional semantics. In 2.2.3 I will describe some basic aspects of quantification in natural language and describe different types of variable binding. I will also discuss Heim's (1982) theory of free variables and Kamp's (1981) theory of discourse markers. Finally I will briefly evaluate the (neo-)
Davidsonian approach of event semantics. This section is not meant as a very brief
summary of semantics. It is only meant as a motivation for the adoption of certain
semantic notions for my theory of negation and Negative Concord.

2.2.1 The Principle of Compositionality

Davidson (1967a) argues that, similar to formal languages, the meaning of an
indicative sentence (a proposition) is captured by its truth conditions: the meaning of a
sentence is constituted by the conditions under which this sentence is true. Hence
Davidson requires that every theory of meaning should minimally fulfil the condition
that the meaning of a sentence follows from its truth conditions. Hence every theory
of meaning minimally requires a truth definition. Davidson follows Tarski (1935,
1956) who argues that a truth-definition should meet a criterion of material adequacy
(14) that he formulates in terms of Convention T.

(14) A truth definition is materially adequate if every equivalence of the form:

\[(T) \quad X \text{ is true if and only if } p,\]

whereby \(X\) is the name of a sentence and \(p\) is the sentence itself or a
translation of it,

can be derived from the theory.

Convention \(T\) is thus not a truth definition, but an instrument to test a given truth
definition on its correctness.

At first sight this may seem trivial, though it is absolutely not, given that this holds for
every sentence in a particular language, including those sentences that have never
been heard or produced before. Speakers of a language understand the meaning of
every sentence in their language, and hence they are able to derive the truth conditions
of all these sentences by means of their parts, i.e. the meaning of all lexical elements
and the way they are structured in the sentence. This asks for a semantic theory that is
based on the so-called Principle of Compositionality.

Gottlob Frege (1892) has been attributed the fatherhood of all semantics theories that
obey the Principle of Compositionality. This principle postulates that meaning is
compositional, i.e. every element that is present in (some parts of) the sentence will
contribute to the meaning of the sentence, and the meaning of a sentence can only be
constructed from the meaning of its parts.

(15) Principle of Compositionality:

The meaning of a sentence is a function of the meanings of its parts and their
mode of composition.\(^{20}\)

This implies that the meaning of a sentence should be constructed from its syntax. Unordered sets of lexical items do not contribute to the meaning of the sentence as the meaning of a sentence is purely reducible from the meanings of the parts and their internal ordering. In the case of unordered sets of lexical items (i.e. without any syntax) the mode of composition remains undetermined and hence the sentence becomes ambiguous (or uninterpretable). Therefore, the input of semantics is a syntactic level of representation. In standard generative theory this level of representation is LF, the interface between syntax and the intentional-conceptual component. One of the main advantages of applying semantics at the level of LF is that it prevents scopal ambiguities that may arise at surface structure. Take for example a sentence as in (16).

\[(16)\] Every boy likes a girl

\[\text{LF1: } \lambda x \exists y [\text{boy}(x) \rightarrow \text{girl}(y) \& \text{like}(x, y)]\]

\[\text{LF2: } \lambda y \exists x [\text{girl}(x) \& \text{every}(x) \rightarrow \text{like}(x, y)]\]

Whereas (16) is an ambiguous sentence, both LF representations are unambiguous and distinct. These kinds of ambiguities were used as motivation for a translation from the surface structure into a logical form before the interpretation procedure takes place. Frege, Russel, Whitehead and Quine, a.o. propagated such theories. Montague (1970) argued that similar to formal languages, natural language should be interpreted directly, i.e. without any intermediate representation, hence introducing Montague Grammar (Montague 1973).

In the next section I will continue the discussion about the division of labour between the syntactic and semantic component.

### 2.2.2 Lambda Calculus and Type-theory

In order to determine the meaning of a sentence as a result of the meaning of its parts, what semantic type each syntactic category corresponds to, should be examined first. There are two basic types of denotations. First, definite DP’s (including proper names) denote entities or individuals. These elements are said to have type \(e\). Second, sentences have type \(t\), since only sentences have a truth-value given their propositional nature. All other categories can be thought of as functions from one type onto another\(^{21}\). These have type \(<\alpha,\beta>\) where \(\alpha\) is a type and \(\beta\) is a type. \(<\alpha,\beta>\) means that it is the type of function that maps elements of type \(\alpha\) onto type \(\beta\). Intransitive verbs map an individual (the subject) to a truth-value (the sentence). Therefore intransitive verbs have type \(<e,t>\). Consequently, transitive verbs have \(<e,<e,t>>\) as

\(^{21}\) This only concerns their basic types. The number of types can be extended to events (section 2.2.4), situations, etc. in order to obtain a more adequate representation of meanings.
they first map an object to an intransitive verb and then a subject to a sentence. Nouns (NP), adjectives (AP's) and predicates (VP's) all have type <e,t>, etc.²²

Functions are formulated by means of lambda calculus. Lambda expressions consist of two parts: the domain of the function (introduced by λ) followed by the value description, i.e. the function over the domain. In the case of an intransitive verb, the domain of the function is D_e, the set of all individuals. Thus λ introduces a variable x of type e. The function consists of the semantic denotation of the verb applied to this variable x. Thus the translation of a verb like 'to sleep' in lambda calculus is:

(17) to sleep $\rightarrow$ λx.sleep(x)

Now we are able to define the interpretation of sentences and parts in terms of functions of elements of different types. The interpretation of an expression of type e is the entity it refers to: the interpretation of the sentence is its truth-value, and the interpretation of any other element is the function it denotes.

(18) a. [[John]] = John
    b. [[snores]] = λx.snore(x)
    c. [[John snores]] = true iff John snores

Let us now look at the interpretation of different parts of the syntactic input for the semantics. One can distinguish terminal nodes, non-branching nodes and branching nodes. Terminals have to be specified in the lexicon; non-branching nodes 'adopt' the interpretation of their daughter; branching nodes are the locus of so-called Functional Application (FA), one of the semantic operations:

(19) a. If α is a terminal node, [[α]] is specified in the lexicon
    b. If α is a non-branching node and β is its daughter, then [[α]] is [[β]]
    c. If {β, γ} is the set of daughters of branching node α, and [[β]] is in D_<e,t> and [[γ]] is in D_α, then [[α]] = [[β]](x) & [[γ]](x)

Functional application is one semantic mechanism. Another semantic mechanism is so-called Predicate Modification (PM). Whenever two daughters of the same projection are both of type <e,t>, e.g. an adjective-noun combination, the interpretation refers to the intersection of the two sets that the predicates refer to.

(20) Predicate Modification²³:
    If {β, γ} is the set of daughters of branching node α, and [[β]] and [[γ]] are both in D_<e,t>, then [[α]] = λx.[[β]](x) & [[γ]](x)

²² Note that this only applies to basic nouns, adjectives or predicates. E.g. the meaning of an intensional adjective like 'possible' cannot be captured by the type <e,t> since it does not map individuals onto propositions.
²³ The term 'predicate modification' is actually a misnomer, as modification implies that an element of a type α is modified by an element of type <α,α>. 
This means that the interpretation of ‘old man’ is a function from entities to truth-values such that the function yields a truth-value 1 if and only if the entity is both in the set OLD and in the set MAN\(^{24}\).

Such semantic theories that use type-theory to compose the meaning of sentences out of their parts are said to be type-driven.

### 2.2.3 Quantifiers and variables

The assumption that definite DP’s such as ‘John’ or ‘the man with the golden gun’ are of type e is appealing. Quantifying DP’s that bind a variable however cannot be regarded as entities as they do not refer to a particular element in \(D_e\). Hence quantifiers are not of type e. As sentences (having type t) can be composed of a quantifier and a predicate (of type \(<e,t>\)) through Function Application (FA), the minimally required type of quantifiers is type \(<<e,t,t>\>\). Quantifiers are thus functions that map predicates onto truth-values. Intuitively this makes lots of sense as expressions like ‘nobody’ or ‘everybody’ tell that no individual respectively every individual has a particular property. Recall that every quantifier has a restrictive clause (such as thing in everything or man in no man walks). Therefore we can assume the following semantics for quantifiers: (21)a-b are members of \(D<<e,t,t>\) but the original quantifier every in (21)c has type \(<<e,t>,<<e,t,t>>\>\) since it maps to predicates of type \(<e,t>\) to a proposition of type t.

\[(21)\]
\[
\text{a. } [[\text{Nothing}]] = \lambda P . \exists x . [\text{Thing'}(x) \land P(x)] \\
\text{b. } [[\text{Somebody}]] = \lambda P . \exists x . [\text{Human'}(x) \land P(x)] \\
\text{c. } [[\text{Every}]] = \lambda P . \lambda Q . \forall x . [P(x) \rightarrow Q(x)]
\]

This analysis that has developed into the theory of Generalised Quantifiers (Montague 1973, Barwise & Cooper 1981) so far only applies to quantifying subjects. Whenever a quantifier is in object position, a type mismatch will occur. In the next subsection I will discuss different kinds of solutions to this type mismatch.

The picture sketched so far shows two kinds of possible treatments for pronominal DP’s: either they are bound variables or they are referring expressions. Examples of the two kinds are found in (22).

\[(22)\]
\[
\text{a. } \text{John likes Mary, He is a fool.} \\
\text{b. } \text{Every boy; says he, likes Mary.}
\]

In (22)a ‘he’ refers to John; in (22)b ‘he’ is bound by the antecedent ‘every boy’. However, it has been observed that some pronouns are neither a bound variable\(^{25}\) nor a referring expression.

---

\(^{24}\) This does not always hold: ‘possible suspect’ is not denoted by the intersection of the set POSSIBLE and the set SUSPECT. Also scalar predicates such as ‘small’ or ‘red’ do not obey predicate modification as defined in (20).
(23)  
\begin{align*}
&\text{a. A man, walks in the park. \textit{He}_i \text{ whistles}} \\
&\text{b. If a man, walks in the park, \textit{he}_i \text{ likes trees}} \\
&\text{c. If a farmer, owns a donkey,\textit{ he}_i \text{ beats it}_j}
\end{align*}

Clearly all pronouns in (23) are not referring expressions as they do not refer to any specific entity. Additionally, they cannot be bound as there is no c-command relation between the indefinite antecedent and the pronoun. Moreover in (23)b-c, the binding relation gets a universal interpretation: ‘every man who walks in the park likes trees’ and ‘every farmer beats all of his donkeys’. Therefore, the semantic theory needs to be expanded\textsuperscript{26}. An approach has been formulated by Kamp and Heim independently: Kamp’s (1981) theory of Discourse Representation Theory and Heim’s (1982) theory of file-change semantics and Heim’s (1990) theory of E-type anaphora. The dynamic Kamp-Heim approach suggests that indefinites are not represented by means of an existential quantifier but as a free variable that is introduced:

(24) $[[\textbf{a man}]] = \text{man}(x)$

All indefinites introduce variables (discourse markers in Kamp’s terms). Definite expressions pick out a referent. Heim captures the representation of a text as a set of file-cards (for every referent one) whereby every indefinite introduces a new file-card and for every definite expression a file-card is updated. Kamp uses Discourse Representation Structures (DRS) to describe the representation of texts in first-order logic using a box notation. These boxes can be seen as a more general form of file-card semantics that does not only include information about referents but also includes conditional and quantificational structures.

Free variables still have to be bound. Given the mechanism of \textit{unselective binding}, any ‘real’ quantifier, including adverbal quantifiers (cf. Diesing 1992), can bind any variable in its scope (i.e. their c-command domain). If at the end of the representation of the text some variables are still free, they will be implicitly bound existentially. This mechanism is called \textit{existential closure}. After the representation of the text has been completed, the truth-conditional interpretation of the text takes place.

The Kamp-Heim approach has led to some unwelcome results, which as a reaction yielded a fruitful branch of semantics working on problems related to so-called ‘donkey anaphora’\textsuperscript{27}. A critical study of dynamic approaches to binding phenomena falls outside the scope of this study. The reason to discuss indefinites in this section is because I will analyze so-called n-words in Negative Concord languages as indefinites bound existentially by a negative operator.

\textsuperscript{26} Note that binding requires a c-command relation.

\textsuperscript{27} The original observation goes back to Lewis (1975). Cf. also Evans (1977) and Cooper (1979).

2.2.4 Event semantics

It was suggested by Davidson (1967b) that action sentences (i.e. non-statives) do not merely express an n-ary relation between the verb’s n arguments, but in fact express an n+1-ary relation between the nominal arguments and an event variable that is bounded existentially. Hence the sentence in (25)a does not get a semantic representation as in (25)b, but as in (25)c.

\[(25)\]
\[
a. \quad \text{Fred buttered the toast slowly in the bathroom with the knife}
\]
\[
b. \quad \text{[with'}(k)(\text{in'}(b)(\text{slowly'}(\text{butter'})))\text{]}(f, t)
\]
\[
c. \quad \exists e[\text{butter'}(e, f, t) \& \text{slowly'}(e) \& \text{in'}(e, b) \& \text{with'}(e, k)]
\]

At first sight, the introduction of an extra ontological category looks superfluous as (25)b and (25)c represent both the interpretation of the sentence correctly. Parsons (1990) however formulates three kinds of arguments in favour of a Davidsonian theory of event semantics: the modifier argument, the argument from explicit event reference and the argument from perception reports\textsuperscript{28}.

The modifier argument compares verbal modification with nominal modification and says that nominal and verbal modification show to a large extent the same kind of behaviour. The main properties of nominal modification are permutation and drop. Permutation says that the internal order of adjectives and adverbial does not matter to the semantics. Drop says that a noun or verb with n+1 modifiers entails the noun or verb with n modifiers\textsuperscript{29}.

This argument faces some serious problems. It has been argued that the internal scope of adverbs is due to a hierarchy (Cinque 1997). This means that the reversion of the order is either impossible or leads to scope changes\textsuperscript{30}. Therefore the semantics of two representations with a different adverbial order are never equal. This claim has been extended to prepositional adverbials (Cf. Koster 1974, 2000, Barbiers 1995, Schweikert t.a.). Also drop leads to problems with modifications by modality adverbs as possibly, probably or perhaps, which certainly do not entail the event. Whereas the first argument introduces unwelcome results, the second argument is more persuasive. Take the following example\textsuperscript{31}.

\[(26)\]
\[
a. \quad \text{In every burning, oxygen is consumed}
\]
\[
b. \quad \text{John burnt wood yesterday}
\]
\[
c. \quad \text{Hence, Oxygen was consumed yesterday}
\]

\textsuperscript{28} Cf. also Landman (2000), a.o. for a support of event semantics.

\textsuperscript{29} This is not the case for all modifiers. Intensional or scalar modifiers cannot always be dropped under entailment. ‘This is a potential problem’ does not entail ‘this is a problem’, and ‘this is a small pink elephant’ does not necessarily entail ‘this is a small elephant’. (Cf. Landman 2000).

\textsuperscript{30} Cinque’s hierarchy excludes permutation of adverbs. However, many instances of reverse adverb orders have been found (Nilsen 2003). These examples do show scopal differences.

\textsuperscript{31} Taken from Parsons 1990.
In this example it is intuitively clear that burning events are quantified over. Rothstein (1995) provides an argument along the same line as Parsons (1990) in favour of the Davidsonian approach.

(27) Every time the bell rings, I open the door

Rothstein argues that there is a matching relation between two event variables in this example, one introduced by *ring* and one by *open*, whereby the ring event requires an event modifier in the main clause. Landman (2000) correctly argues that the argument is convincing with respect to the ring event but does not prove that there is a distinct opening event as the event argument could also be introduced by the event modifier through an instantiation relation. Thus, the argument from explicit event reference proves the possibility of the introduction of an event variable by a verb but not its necessity.

Finally, the argument from perception reports (as (28)) is in favour of the Davidsonian approach since it makes clear what the theme of *saw* in (28)a is, or, what *it* in (28)b refers to.

(28) a. John saw Mary leave
   b. John was in love with Mary. It made her unhappy.

Thus it is shown that there is good evidence to assume event variables in some cases, but not in every case. Therefore, it does not sound plausible to assume that the event variable is introduced by every verb in every situation\(^\text{32}\). Analyses in which a syntactic category that is distinct from the verb (e.g. \(v\)) introduces an event variable are more attractive in this way (Cf. Chomsky 1995, Ernst 2001, Pylkkänen 2002).\(^\text{33}\)

It would be presumptuous to evaluate the discussion and literature on event semantics in this small subsection and this is not my intention either. This subsection is simply meant to show that the assumption of an event variable in a semantic representation is plausible, without making claims about the necessity of event semantics in general or any specific theory in particular. Basically this is what I need for my analysis of negation, as I will show that in some languages sentential negation involves binding of events by a negative operator.

This section on semantics ends here. In a brief description I discussed the main theories and tools that I will need for my analysis of negation. A detailed semantic analysis of negation will be presented in chapter 7 and 8.

---

\(^{32}\) This is not Davidson’s claim either, as he restricts the introduction of events to action sentences.

\(^{33}\) Or, in a Larsonian V shell structure, the highest instance of V.
2.3 The syntax-semantics interface

In the previous sections I described two different domains of linguistic theory. In this section I will describe the relation between these two linguistic fields: the syntax-semantics interface. This interface is subject to the study of how interpretation follows from a structured sentence. As has been shown in the previous section, semantics is derived from the syntactic structure that forms its input. The question in this section is to determine how, where and when syntax meets semantics. In other words, which part of the interpretation is the result of the syntactic derivation and which part of the interpretation is the result of the semantic mechanisms.

Contrary to the previous two sections, which were merely descriptive in nature, this section tries to bridge two sometimes conflicting linguistic fields: One of the central questions in the study of the syntax-semantics interface is the determination of the borderline between syntax and semantics, i.e. where does syntax stop and where does semantics start. Whereas several radical syntactic theories try to reduce semantics to an application of syntactic principles, some semantic theories take syntax to be nothing more than a categorical grammar underlying the semantics, leaving no space for a particular syntax of natural language.

The fact that some of the syntax-semantics interface theories are in conflict with each other requires a less objective evaluation, as the theory of negation that I will present demands a coherent and consistent vision on the interplay between syntax and semantics. In this section I will discuss different proposals concerning the relation between syntax and semantics. In the end, I will propose to take Spell-Out as the border between the two disciplines, although the exact location of this border will turn out not to be crucial for my theory of negation, i.e. it will be applicable in multiple frameworks regarding the syntax-semantics interface.

In the first subsection I will discuss scope ambiguities, in order to show that syntactic surface structures do not always correspond to the semantic input that the correct interpretation requires. In the second subsection I will broaden the topic to the general question where interpretation takes place: at surface structure or at LF.

2.3.1 Scope ambiguities

In 2.2.3, I discussed that quantifiers are of type $<e,t>,<e,t>,t>$ and that quantifying DP's are of type $<e,t>,t>$. This led to the correct semantics for quantifying subjects, but still leads to a type mismatch for quantifying objects. The VP in (29) for example does not allow for FA.

(29) $[[\text{DP } \text{Mary} ] [\text{VP likes someone}]]$
    $[\lambda y.\lambda x.\text{like}'(x,y))(\lambda P.\exists x.[\text{human}'(x) & P(x)]])$

Clearly the object cannot apply FA over the verb because the transitive verb is of type $<e,e,t>>$. But the verb obviously cannot take the quantifier as its argument either.
The problem becomes even more complex when we substitute the subject with another quantifier, since such a sentence does not have one possible interpretation, but two.

\[(30) \quad \left[\left[ DP \text{ Everybody} \right] \left[ VP \text{ likes someone} \right] \right]
\]
\[
\forall x [human'(x) \rightarrow \exists y [human'(y) \& like(x,y)]]
\]
\[
\exists x [human'(x) \& \forall y [human'(y) \rightarrow like(x,y)]]
\]

In one reading the subject scopes over the object, and in the other reading the object has scope over the subject. This leads to the questions, what is the correct interpretation procedure of non-subject quantifying DP’s and how does one account for these so-called scope ambiguities. This question has been puzzling syntacticians and semanticists for the last decades, leading to several (kinds of) approaches. Basically there two ways of tackling the problem: either there is an abstract syntactic structure (an LF) for every possible interpretation, in which the type of the quantifier object does not cause type mismatches, or the interpretation does take place at surface structure, with type-shift operations being responsible for the correct semantics. In the rest of this section, I will briefly describe the two approaches.

The first approach uses movement as a solution. Although Montague’s system tries to derive interpretations at the surface level, his solution contains some additional syntax only motivated for interpretation reasons. His solution, quantifying-in\(^{34}\), replaces the object quantifier by a pronoun-trace \(x_n\) that will be bound by a lambda operator in a later stage of the derivation so that FA becomes possible. The subject-predicate yields a proposition \(p\) of type \(t\), but as the pronoun \(x_n\) needs to be abstracted, the result \(\lambda x_n.p(x_n)\) is of type \(<e,t>\). The object quantifier takes this as its argument, yielding a proposition with the object scopin over the subject. In a proposition with multiple quantifiers, different structures are constructed, each leading to different orderings of quantifiers, in order to derive the possible scope ambiguities. Note that the input for semantics is still the surface structure, but that the types of the quantifiers trigger additional syntax to complete the interpretation procedure. Rather than taking the surface structure as input for semantics, May (1977) argues that syntax does not stop at Spell-Out but continues the derivation until LF has been produced, a level on which interpretation takes place, i.e. the input for semantics. Quantifiers move covertly to a position from which they take scope (in terms of c-command). This mechanism is called Quantifier Raising (QR) and has been adopted in several modified versions (Huang 1982, May 1985, Fox 2002, Reinhart 1995). Recently, Beghelli & Stowell (1997) have developed a minimalist version of QR in terms of feature checking and quantifiers take scope from different positions in a fine-grained split-CP structure.

Chapter 2 - Theoretical backgrounds

(31) SO \[[\text{IP Everybody loves someone}]\]
LF \[[\text{IP Somebody [Everybody [loves somebody]]}]\]
\[\forall x[\text{hum}(x) \rightarrow \exists y[\text{hum}(y) & \text{like}(x,y)]]\]
\[\exists x[\text{hum}(x) & \forall y[\text{hum}(y) \rightarrow \text{like}(x,y)]]\]

The second approach tries to derive the correct interpretation without assuming extra syntactic structure that is purely motivated for interpretation reasons. One proposal formulated by Cooper (1975, 1983) is quantifier storage (also known as Cooper storage). Cooper storage takes the interpretation of a DP not to be atomic, but forms an ordered set of different interpretations, of which the first member is the traditional denotation and the ‘stored’ ones are those that are required to obtain scope from a higher position by means of quantifying in. Inverse subject-object scopes are thus obtained by first deriving the subject-object reading and then ‘taking the object out of storage’ and having it take scope on top of the subject. The translation of every DP is a set consisting of a basic interpretation and a sequence of ‘store interpretations,’ which can be taken out of storage later on by means of quantifying-in. The interpretation (32) of can then be derived.

(32) Every man finds a girl

The basic translations for every man and a girl are:

(33) a. Every man \( \rightarrow \{<\lambda P.\forall x[\text{Man}(x) \rightarrow P(x)]>\}\)
b. A girl \( \rightarrow \{<\lambda Q.\exists y[\text{Girl}(y) & Q(y)]>\}\)

DP storage yields an infinite number of non-basic translations

(34) a. \( \{<\lambda P. P(x_i)>, <\lambda P.\forall x[\text{Man}(x) \rightarrow P(x)], x_i >> | x_i \in N\} \)
b. \( \{<\lambda Q. Q(x_i)>, <\lambda Q.\exists y[\text{Girl}(y) & Q(y)], x_i >> | x_i \in N\} \)

A possible subset of the meaning of (32), consisting of the basic translation for the subject DP and a stored interpretation for the object DP is:

(35) \( \{<\forall x[\text{Man}(x) \rightarrow \text{finds}(x, x_i)]>, <\lambda Q.\exists y[\text{Girl}(y) & Q(y)]> | x_i \in N \} \)

Now the final step is NP retrieval in which the second sequence is applied over the first by means of quantifying-in over \( x_i \).

(36) \(<\lambda Q.\exists y[\text{Girl}(y) & Q(y)](\lambda x_i.\forall x[\text{Man}(x) \rightarrow \text{finds}(x, x_i)])> = \}<\exists y[\text{Girl}(y) & \forall x[\text{Man}(x) \rightarrow \text{finds}(x, y)]>\)

This yields the object > subject reading from the same syntactic structure as the subject > object reading is derived.\(^{35}\)

\(^{35}\) Examples taken from Hendriks (1993).
Although this simplifies the required syntax to only one structure, it makes the interpretation procedure far more complex, as the interpretation does not yield sets of meanings (for ambiguity) but "sets of sequences of sequences of meanings"\(^{36}\).

In order to prevent this additional machinery, Hendriks (1993) proposes to loosen the strict correspondence between syntactic categories and semantic types and accounts for scopal ambiguities using flexible types, in which the semantic type can be lifted to a more complex type if the scope of a quantifier so requires. The idea is in a way similar to Cooper's idea that DP's have a basic translation and a 'higher' translation that accounts for the scope effect. However, Hendriks' system is able to derive the proper readings without adding as much new machinery to the semantics as Cooper's system does.\(^{37}\)

In this subsection I discussed very briefly the two approaches of scope ambiguities: by means of QR and by assuming (type) flexibility of the semantics of quantifiers. Note that both approaches are not necessarily restricted to quantifying DP's, although these may be the most common quantifiers. Adverbs can be regarded as (generalised) quantifiers\(^{38}\) as they bind variables too, such as temporal adverbs binding time variables. Likewise, negation can be seen as a quantifier that binds events.

Moreover, sentences consisting of a quantifying DP and a negation can (in some languages) also give rise to ambiguity.

\[
\begin{align*}
(37) & \quad \text{Every man didn't leave:} \\
& \quad A > \neg \, '\text{Nobody left}' \\
& \quad \neg > \forall \, '\text{Not everybody left}'
\end{align*}
\]

One strategy for solving this ambiguity is either by assuming lowering of the quantifying DP (reconstruction) or by assuming raising of the negation to a higher position (neg-raising). The analysis of neg-raising became popular because moving negation to a higher position turns the negative operator into an operator that maps (positive) propositions into (negative) propositions (hence of type \(<t,t>\)). In a higher position the negation could simply take its propositional sister as its complement. Another strategy, however, would be to think of negation in terms of an operator with a flexible type. In that case, depending on the type of the negative operator, the correct reading can be produced. In chapter 6 and chapter 8, I will discuss the properties of the negative operator in detail, arguing for a flexible negative operator that can apply to elements of different semantic types.

\(^{37}\) It should be noted that Hendriks (1993, ch. 2) places this machinery in the syntactic component again in order to avoid problems with respect to compositionality. Hence this mechanism is not entirely semantic.
2.3.2 The level of interpretation

The question where syntax meets semantics has been a long debate that probably will never end. One of the two main approaches says that syntax constructs Logical Forms that are unambiguous, from which the semantics then gives the correct interpretation. Ambiguity is then the result of two multiple LF’s that are derived from the same syntactic structure (or linear order) at Spell Out. The other approach takes interpretation to take place at surface structure and uses semantic devices to derive the correct interpretations. This does not exclude that there are no ways in between. It might be the case that some phenomena are derived by QR and others by semantic mechanisms. Another alternative would be to assume that the interpretation and the syntactic derivation go hand in hand and that semantics does not take completed syntactic objects as its input. Note that this falls back to the original Montague approach but that in a way this is also a radicalisation of multiple Spell-Out in the sense of Epstein & Seely (2002b).\(^{39}\)

The debate about the locus of interpretation has to a large extent a conceptual and theoretical nature, and the two approaches seem complementary in terms of theoretical complexity: a reduction of the theory of syntax as a result of deleting QR leads to more complexity in the semantic system and vice versa. Therefore heuristic arguments rather than empirical arguments have been produced many times for particular analyses. Yet, it is not impossible to produce empirical arguments in favour of LF movement: it has been argued that Antecedent Contained Deletion (ACD) requires movement of the quantifier out of the subordinate clause to an adjunct position on TP, leading to an LF as in (38).

(38) a. I read every book that you did
b. \([[[\text{DP Every book}_1 \text{ that you did read } t_1]]][\text{TP I [PAST read } t_1]]\)

If the quantifying DP has moved out of its low position in the subordinate clause, it leaves a trace that is identical to the trace in the main clause. Therefore the VP’s in both the subordinate clause and in the main clause are identical. This identity licenses the VP deletion in the subordinate clause.\(^{40,41}\)

Other arguments are that QR is clause bound and that QR interacts with other phenomena such as VP-ellipsis and overt Wh-movement\(^{42}\), which are all sensitive to syntactic effects such as cross-over. In (39) the object quantifier may not cross over the c-commanding pronoun he, but if the pronoun does not stand in a c-commanding

\(^{39}\) This is also in line with Jacobson’s (1999) ideas on variable-free semantics and direct compositionality.

\(^{40}\) This argument goes back to Sag (1980) and is repeated in May (1985).

\(^{41}\) See Jacobson (1992) for an account of ACD without LF movement.

\(^{42}\) For more examples, see Szabolcsi 2001.
relationship with the object, as holds for the possessive pronoun *his*, LF movement is marginally acceptable.

(39) a. He admires every man

\[ \forall x. [\text{admire}(x,x)] \]

b. His mother admires every man

\[ \forall x. [\text{admire}(x's\ mother,x)] \]

Note also that QR is also taken to be responsible for the covert movement of *Wh*-words in *Wh*-in situ languages such as Chinese. Also in those cases, QR is sensitive to phenomena such as island effects, e.g. *Wh* islands.

(40) a. \*Who are you wondering whether to invite

b. *How are you wondering whether to behave

(41) a. Ni renwei [Lisi yinggai zenmeyang chuli zhe-jian shi] \(^{13}\) Chinese

You think Lisi should how handle-this.CL matter

‘How(manner) do you think Lisi should handle this matter’

b. \*Ni xiang-zhidaou [shei zenmeyang chuli zhe-jian shi]

You wonder who how handle-this.CL matter

‘How(manner) do you wonder who handled this matter’

However, other analyses for these phenomena are not inconceivable. The fact that current analyses deal with these problems in a syntactic fashion does not exclude or disprove the possibility of formulating semantic answers to these problems. *Wh* island effects for example can be thought of as semantic effects because they are all introduced by downward entailing contexts\(^ {44}\). Clause-boundedness and the cross-over effects may perhaps also be accounted for in a semantic way.

The arguments against LF movement are primarily conceptual. Note that LF movement is a stipulated notion after all as evidence can only be proven indirectly. Therefore LF met much sceptis. I already presented some of the semantic solutions (Cooper and Hendriks), but also the status of LF movement within the syntactic program is still unclear\(^ {45}\). Williams (1986, 1988) is the first to argue against LF, and he suggests that the function of LF should be divided amongst other components of the system. Hornstein (1995) argues that scope effects are due to deletion of copies of elements in A-Chains before meeting the CI-interface. Since the choice of deletion is free in A-Chains, scope ambiguities are expected to occur. Kayne (1998) shows that subsequent remnant movement can account for many scope ambiguities. The underlying idea is that both quantifiers move out of the VP in order to check their

\(^{43}\) Data from Legendre, e.a. (1995).


\(^{45}\) Note that the minimalist model of grammar depends on the notion of LF. However, erasure of LF in this sense would not undermine to the model, only movement between Spell-Out and LF would be forbidden, so that the level of interpretation is surface structure.
features and that the VP is fronted afterwards. This system could then produce surface structures as in (42), in which each surface structure is unambiguous. Syntactic proposals like this one can provide the correct semantic inputs at surface structure.

\[ \text{(42)} \quad \text{I force you to marry nobody} \]
\[ \text{I force you to [marry, [nobody, [t, t_j]]]} \]
\[ \text{I [force you to marry], [nobody, [t, t_j]]} \]

An alternative way of discarding LF movement by movement before Spell-Out is by assuming that covert movement is nothing but feature movement and that therefore all movement takes place before Spell-Out. As the minimalist system provides this kind of movement, LF movement is no longer necessary and can therefore on minimalist grounds be ruled out.

To conclude, there is hardly any empirical support to maintain the notion of LF-movement, but there is hardly any counter-evidence either. Still, empirical arguments should form the basis for the adoption of a level of interpretation after Spell-Out. Several empirical phenomena have led to an explanation in terms of LF movement: Quantifier Raising phenomena, such as inverse scope effects, ACD and (weak) crossover effects; reconstruction (in which an element lowers at LF to yield the correct interpretation) and neg-raising effects (in which the negation moves to a higher position in the clause).

In this study I will remain professionally impartial with respect to the question whether LF should be adopted as a separate level for interpretation, or whether semantic mechanisms like type shifting should be introduced. An elaborate analysis of these phenomena is beyond the scope of this study. Therefore I will restrict myself only to the role of negation as neg-raising as an argument in the debate, and I will show that negation does not provide any arguments in favour of the adoption of LF as a separate level of interpretation by showing that all interpretations of sentences in which a negation has been included can be derived from surface structure.

### 2.4 Conclusions

In this chapter I have provided a brief overview of current syntactic and semantic theories and assumptions and their internal relationships. The first motivation for is to describe the tools and machinery that I will be using throughout the rest of this book. The second motivation concerns the interface between syntax and semantics and the question whether my theory of negation provides arguments in favour of LF movement. In this study I will develop a syntactic model in which interpretation of negation takes place at surface structure. As a consequence, neg-raising, which has often been presented as an argument together with QR, in favour of LF movement, does not provide evidence for the assumption of a level of interpretation that differs from surface structure. I will show that all correct interpretations can be derived from surface structure. Thus I bring a contribution to the debate about the locus of
interpretation, arguing that negation cannot be a motivation to adopt a level of Logical Form after Spell Out.
In the rest of the book, I will combine minimalist syntax and truth-conditional semantics, including indefinites in the Heimian sense, and I will adopt the notion of events without making any further claims about their status in the semantic theory.