4. GRAMMAR: the reward of incompleteness

4.1 THE THREE DUALS OF GRAMMAR

The insight implicitly emerging from the previous chapters is that structure and meaning are non-compliant in a fundamental and principal way. The insight lies sheltered under the idea that, somehow, interpretation of well-formed sentences and assignment of well-formed sentences to given meanings must be computable, even if the ultimate semantic representation is no longer the result of derivation. The present chapter, however, sails under a different perspective. Here, we try to find out what the overall characteristics of the resulting formal system turn out to be. Here, grammar will be identified as incomplete *per se*, in a logical, almost Gödelian sense. Logical incompleteness will be acknowledged as a valuable feature of formal grammar, and as a mark of soundness. In order to ensure computability however, the lexicon will once more be constructed as a treasury of oracles that connect structures to meanings. As a main line, this chapter argues that constituents cannot derive entailments, or the other way round. Hence, some true statements about the relationship between a structured sentence and its interpretation cannot be proven within grammar. This particular gap between truisms about form and meaning and theorems on this relationship constitutes the incompleteness of the grammar. Moreover, the chapter says that logical incompleteness is a desirable property of formal grammar. Grammars that do not enjoy incompleteness are bound to underspecify syntax, semantics, or both.

In daily linguistics, syntax and semantics are different games, to almost every grammarian. Across frameworks, and also in the DELILAH system, syntax identifies and manipulates *constituents*. Constituents are the sentence's proper parts and are taken to be complex bundles of properties – they are complex

symbols or signs. Their interaction is modeled by a certain comparison of these bundles: *unification*. Sentential semantics is about the identification of *entailments*, or semantic consequences – Aristotle's heritage. The semantic contributions of constituents are taken to be functions; they are *composed* in order to provide the logical base for entailments.

Talking about a grammar as a whole, we have to deal with three duals. Firstly, we have to account for the balance between syntax and semantics, the *arts* of grammar. Secondly, we have to compare the *modes of operation* of syntax and semantics, to wit, unification of signs and composition of functions, respectively. Thirdly, we must evaluate the relationship between the *objects* of syntax and semantics: constituents and entailments. The present chapter is on the clash between the arts, the operation modes and the objects of grammar as a formal system. There is no doubt, however, that the theatre of the clash is restricted to grammar, and does not extend to human language.

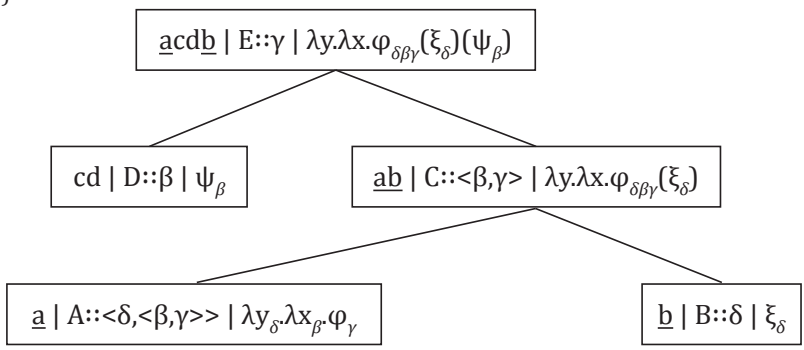
4.2 THE CONSERVATIVITY OF SYNTAX

4.2.1 The yield of syntax

Syntax serves a few good purposes. Firstly, it defines the proper parts of well-formed phrases, marking their roles: the *constituents*. Undoubtedly, they are the keys to grasping language. Secondly, syntax labels constituents, by classifying constituents into equivalence classes, the *categories*. Thirdly, syntax provides the *linearization* regime, inevitable since languages consist only of meaningful sequences of symbols. And, finally, syntax provides some grouping or storage of semantic terms: an *underspecified semantic representation* (cf. Bunt 2008). Below, you will find a picture of a syntactical derivation, which sketches the simultaneous computation of strings, categories and types and meanings.

For rules of grammar C *concatenates* A *and* B and E *infixes* D *in* C and a data triple *<string | category::type| meaning>*, the derivation of a discontinuous string *acdb* of category E is schematized by

(422)



In every step in the derivation, linear order, categories and types and storage of meaning are taken care of. This is why syntax deserves our permanent affection and attention.

4.2.2 The restrictedness of unification

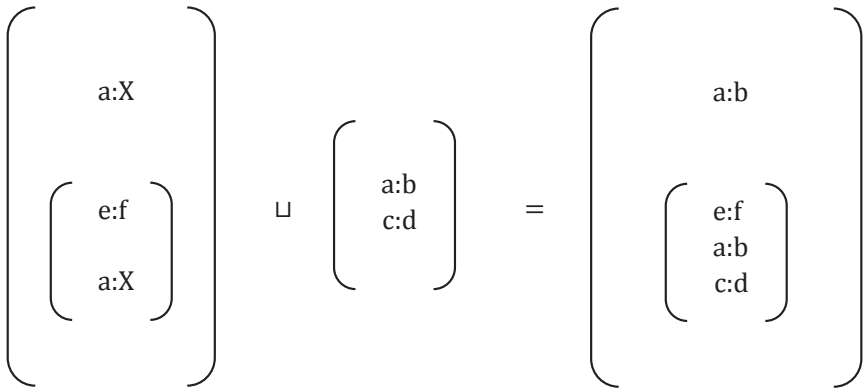
Syntax determines constituency by unifying complex symbols or signs. Unification is a well-defined operation (see *e.g.* Pereira and Shieber 1987) checking the compatibility of graphs or terms by seeking to create one graph or term out of two. Without loss of generality, below, we restrict our attention to graphs.
In the grammar of natural language, applying unification makes sense if four conditions apply:

(a) *binarity:* unification is an operation involving two graphs, yielding a derived graph in case of success
(b) *locality:* everything that is worth unifying is specified in the operanda
(c) *antisymmetry or strong typing:* for each two signs, there is a unique way of integrating two signs
(d) *recursion:* the output of one round of unification is the input of another.

Binarity is an economic principle. The unification of more than a pair of graphs is costly and possibly undecidable. Nothing in natural language calls for such a complication. Locality has to guarantee that all relevant information is available and can be exploited at crucial moments in the derivation of the covering structure. It is a principle of information transfer. Antisymmetry ensures different roles for each graph in the unification. These roles are in the graphs themselves, as part of the specification; therefore, these graphs are strongly typed: the type of a graph determines its role in a unification check-up. Recursion, finally, expresses that unification is all there is to syntax – the outcome

of one unification is the input of another instance of the same process. Below, the process is illustrated as the generalization of two structures; the process was discussed extensively in chapter 3 (cf. (395)).

(423) *Unification of complex signs*



One structure checks another by binarity and antisymmetry. The checking graph has a relevant substructure which is specified for features *e* and *a*. The first has value *f*, the second is undetermined or variable. The value of *a* in the substructure is bound to the value of *a* at the top level of the checking structure; this relationship reflects locality. The secondary structure – the one to be checked – is specified for features *a* and *c,* with definite values *b* and *d* respectively. The specifications for the substructure and the secondary structure are compatible: *e:f* and *c:d* do not affect each other, both occurring in just one sign, and the value X of *a* in the checking sign is instantiated by *b* in the secondary sign. Because no incompatibilities arise, the resulting unified structure specifies all information in one new graph, which is open to unification with other graphs, by recursion. The resulting structure is at least as specific as either of the two operands.

As described here, unification is a standard tool in all lexicalist approaches to grammar (*e.g.* Sag, Wasow and Bender 2003, Bouma 1993, Kay and Fillmore 1999). These approaches differ mainly as to the embedding and organization of unification and – of course – the nature and the logic of the features specified. Unification, as Sag *et al.* note, is not the theory of grammar – it is its chariot.

4.2.3 The generality of unification

Syntax is both constructive and conservative. It specifies how phrases combine to make new phrases in such a way that the phrases entering the combination survive as proper parts in the whole. In syntax, phrases are preserved under combinatorics. Syntax does not delete – it builds. For that reason, the logic of categorial grammar, for example, is intuistionistic, constructionist and resource-sensitive (Van Benthem 1991). For that reason, syntax does not have a complement type of operation. One cannot nullify structure.

Combining phrases syntactically amounts to checking their properties, finding some linearization and computing the properties of the whole out of the properties of the proper parts. It is an essential feature of all major syntactical frameworks that the combinatorics does not create structure independent of information carried by the operands. This is even true for those syntaxes that embody theories of language, like minimalism (Chomsky 1995). The definition of its single (or main) syntactical process *merge* is like (424) rather than like (425), according to Chomsky (1995), Collins and Stabler (2011) and Stabler (2010).

(424) merge(α, β) = {β, {α, β}}
(425) merge(α, β) = {γ, {α, β}}.

The result of combining two things syntactically is projected from the objects combined, to wit, α and β. The resulting structure is not projected out of the blue. It is not surprising, then, that independently formulated phrase structure grammars with production rules have an equivalent formulation in a more lexicalist framework, like categorial type logic: if a category A expands to B and C according to some phrase structure grammar, one can equivalently state that members of category B map category C to category A, making the category B combinatorially obsolete. Phrase structure rule (426) expresses the very same syntactical information expressed by functional application rule (427).

(426) φ A ψ → φ B C ψ
(427) φ A ψ ⇐ φ A/C C ψ

In (426), the top label A is not necessarily associated with the daughters' labels. In (427), the top label is introduced by one of the daughters, by definition. The relation between phrase structure grammars and categorial or type-logical grammars has been studied since Bar-Hillel (1953). Pentus (1993) proved the (weak) equivalence of context-free phrase structure grammar and

the Lambek-calculus – restrictions of type-0 grammars and of type-logical grammars respectively. The mapping between minimalistic and categorial grammar has been studied by Retoré and Stabler (2004) and Vermaat (1999). What matters here is that in (424) and (427), the top node – the new constituent, the unification, the whole – does *not* introduce new information. The labeling of the new constituent comes with its proper parts.

Grammars do not describe the manipulation of categories, though, but the manipulation of richly structured phrases, containing all kinds of information. Let us call these phrases of a certain category *complex symbols* or *signs*, as was suggested by, among others, Friedman and Bredt (1968), Gazdar, Klein, Pullum and Sag (1985), Morrill (1994), and many others. Under that view, common to almost all present approaches to syntax, rules of grammar with formalizations like (424) or (427) specify how complex symbols arise from other complex symbols. Linguistic theories differ considerably – though not fundamentally – in the specification of complex symbols, as to the nature and the amount of information stored in the complex symbols, and as to the alleged impact of the representations. Discourse representation theory (Kamp and Reyle 1993), for example, is much more specific on anaphoric effects than are other approaches to grammar. Yet, the basic process that lives on all these resources is *unification.* Unification is a costly, but finite procedure to check point by point whether the requirements that one constituent imposes on its linguistic environment are met by candidates for that environment. The division of labour between the signs is that one sign specifies conditions for the other, while the other specifies values in the checking sign. If all relevant specifications match point by point, the unification succeeds. If one specification runs counter to the other, unification fails. If specifications do not interfere, they go along with the process. Information is neither destroyed, nor left behind. At most, some general information is replaced by more specific information of the same sort. In particular, if A is a constituent of B and unified with it, subsequent unification of B with another constituent *cannot* destroy the position of A with respect to B. Essentially, the unification of A and B is at least as informative or specific as both components; it is conservative.

Note that unification does not impose any restriction on the nature of the constraints. It is important to see that even semantic information can be checked and transferred this way. However, the level of semantic representation that can be reached by applying unification is by conjecture exactly the level that is called *underspecified semantics* in computation (Bunt 2008). It is a kind of storage of typed semantic contributions where the ultimate inter-

action between these contributions is not specified. Similar approaches to delayed application have been proposed by Cooper (1983), Janssen (1986), Bos (2001), Copestake *et al.* (2005) and Jäger (2005).

Syntax is conservative, because all its aims can be fully achieved by unification and unification does not destroy any information. Successful unification yields a conservative merge of complex symbols. Information is added to other information; no information is lost. Not all information will be available at all levels of embedding, but that is not the point: information is not expelled by other information, and the level of specificity is monotonously increasing: with each unification new information comes in. For this reason, we can call syntax an *additive* process. It is the grammatical instantiation of addition and union. It defines an algebra in which the smaller units are ordered below the larger ones. If two complex signs unify, both define proper constituents of the resulting phrase. In the following, the symbol $\sqsubseteq$ reads as 'is a constituent of '; this notion is antisymmetric.

(428) If A and B are complex signs and $A \sqcup B$ is their unification, then both $A \sqsubseteq A \sqcup B$ and $B \sqsubseteq A \sqcup B$.

Lemma (428) we call *additivity* for syntax, as it compares syntax to set union and partial (transitive, antisymmetric) ordering over sets, as well as to addition over $\mathbb{R}+$ and its ordering:

(429) If A and B are positive real numbers and A + B is their addition, then both $A \leq A+B$ and $B \leq A+B$.
If A and B are sets and $A \cup B$ is their union, then both $A \subseteq A \cup B$ and $B \subseteq A \cup B$.

In additivity for syntax, the empty complex symbol is the unit, but the grammar will never schedule the empty complex symbol for unification, as it does not represent any syntactical object. The syntactical representation of a sentence with constituents A_1 .... A_n is therefore $(...(A_1 \sqcup A_2) \sqcup ..... \sqcup A_n)$ or $\coprod_{i=1..n} A_i$. It represents a conservative and preserving merge of information, scheduled by rules of grammar.

4.3 THE DESTRUCTIVITY OF SEMANTICS

4.3.1 Divorcing constituents and entailments

Natural language semantics west of the Indus starts with Aristotle. The *Organon* describes how two independent sentences logically determine a third one. Here is, as an example, the syllogism dubbed *Cesaro,* combining a negative (e), a universal (a) and an existential (o) quantifier*.* A negated existentially quantified sentence and a universal quantified sentence jointly entail an existentially quantified sentence with predicate negation. The conclusion takes its predicate negation from the negative quantifier in the first premise, its left predicate or middle term from the second premise, the positive version of its right predicate from the first premise, and its quantifier out of the syntactical blue. The abstract syntactical structure in Germanic languages for this syllogism is indicated below.

(430)	No humans have hooves	$[_{dp}$ Q2 P] M
	All horses have hooves	$[_{dp}$ Q1 S] M
	∴ Some horses are not humans	$[_{dp}$ Q3 S] Q2-P

The *conclusion* is entailed by the premises – one cannot consistently assert the premises and deny the conclusion. That is, all speakers of a language will confirm that they cannot think of a situation in which the following complex description in their language would be true, that is, in which it could be anything other than a desperate expression of nonsense.

(431) No humans have hooves and all horses have hooves but some horses are humans.

Natural languages – even Pirahã, we must assume – share these syllogistic three-tuples, because all languages have the partial ordering coming with a universal quantifier (Zwarts 1983) and they have negation, and that should do. In most languages, these quantifiers are conservative, restricting the domain of quantification to the nominal property, and go with that property syntactically, rather than bridging between the expressions of that property. And in all languages, the single conclusion is built from several syntactically specified but unrelated constituents. Despite all mediaeval classifications of syllogisms in term of subjects, predicates and midterms, however, Aristotle's is not a theory on the syntax-semantics interface. From the constituent structure of the premises one cannot predict the conclusion. From the conclusion

one cannot deduce the structure of the premises. The relationship between constituency and entailment is very loose. It is not possible to map constituents and entailments functionally.
From its very inception as a field of empirical study, semantics went beyond syntax.

4.3.2 Entailments as products of composition and deduction

Semantics is invasive and destructive, and its operations may not even be computable (Lasersohn 2009). The semantic representation of a sentence has to take care of all those properties of a sentence that are not determined, or are under-determined, by its structure. It does so by bringing certain terms under control of operators outside that term, thereby affecting the independence of the semantic contribution of the term *per se*. The intensional independence of the term is reduced to denotational dependency when the term is absorbed in the proposition. That is what the 'extra-grammatical' meaning postulates of Montague (1971) are about. The destructive nature of semantic composition emanates in two related but distinguishable phenomena: scopal control – affecting the interpretation of operator-sensitive terms – and phrasal or structural ambiguity – the fact that one syntactical structure may come with several distinct interpretations. Scopal control changes the way in which a phrase contributes to the meaning of the whole. Ambiguity of the kind that occurs in many *extended lexical units* where a certain grouping may or may not introduce a specialized reading can overrule the individual lexical elements completely: in one reading of *John kicked the bucket,* there is neither a bucket, nor does any kicking occur. To the extent that syntactical structure and categorization is unaffected by the outcome of the interpretation, scopal control and ambiguity turn semantic terms into dynamic warriors, striving after reign instead of just preserving *part-of*-ness, as in syntax. If two terms φ and ψ each contain an operator, and if these two operators can dominate each other, the possible relations between the operators have to be checked when φ and ψ are applied to each other. If a phrase [A B] in *X A B Y* can contribute, for example, the application of the meaning of A to the meaning of B, it has to be checked whether it had not rather contribute some other meaning, not arising from functional application. A classic example here is the *way* construction (Goldberg 1995). In Dutch (cf. Poß 2010), intransitive verbs can productively introduce a causative construction when combined with expressions of path and direction and a reflexive – this construction was already mentioned in sections 3.1.4 and 3.4.4. Informally, a Dutch intransitive verb

V creates a fully-fledged paradigm {*V+reflex+een weg naar DP}*, meaning: by V-ing achieve to end up in DP. Here, we repeat a slightly modified example from chapter 3, with intransitive *golfen* 'play golf' as verbal head.

(432) Elke bankier had *zich een weg naar* een Raad van Bestuur kunnen *golfen*
Every banker had himself a way to some Board of Directors been-able play-golf
'Every banker could have succeeded in moving into some Board of Directors by playing golf'

Note that instead of *golfen,* any other intransitive verb could occur, maintaining the interpretative framework. In this sentence, *[een weg naar een Raad van Bestuur]* is a constituent. Whether and, if yes, how this constituent contributes to the meaning of the whole, and also whether both occurrences of *een* are interpreted as quantifiers at all, entertaining scopal relations with the universal quantifier and modal, for example, completely depends on the details of the construction's embedding, and the way the relevant meanings are stored.
The interpretation of a sentence is both kingmaker and exterminator. The semantic representation of a sentence – its Logical Form (LF) – is accountable for all and only its *entailments*. Among grammarians, this concept of Logical Form has already been revealed in Higginbotham (1985). Entailments are the logical consequences of a sentence, that is: those consequences that are independent of the context of utterance or knowledge of the world. That is why they live in the domain of grammar. Entailments are intensionalized or necessitated implications, induced by knowledge of language. Aristotle's syllogisms are the prototypical examples of entailments. The semantic representation of *All human beings are mortal* and *Socrates is a human being* must be such that from their conjunction the proposition *Socrates is mortal* can be derived with purely formal, logical manipulations. For a sentence like (433), the semantic representation must be such that each of the independent sentences in (434) can be derived from it, by logical manoeuvres and grammar alone. At the same time, the semantic representation or logical form must resist the deduction of any of the sentences in (435).

(433) Not every farmer beat his donkey
(434) Some farmer owned a donkey
Some donkey was not beaten
Some donkey that relates to a farmer was not beaten
Some farmer did not beat a donkey that was related to him
Someone did not beat
Some creature was not beaten
Some donkey was not beaten by any farmer to whom it relates
...

(435) No donkey was beaten
Every donkey was beaten
Some donkey was not beaten by any farmer
…

The sentences in (434) are simultaneous consequences of (433). Of course, these consequences are relative to some well-defined calculus of propositions. To be a semantic consequence is to be derivable by some logic. The formal tendency in modern semantics is motivated only by the fact that formal logics and algebras provide mechanic deduction: the theory of grammar needs these mechanics to justify the inevitability of entailments observed by Aristotle. Therefore, the logical form of a sentence can be defined with respect to its entailments. Its ideology is captured by the following statements, requesting some consistent logic.

(436) *Logical Form of S*
The logical form of a sentence S is that representation ⟦S⟧ for which a logic L exists such that S entails P iff $⟦S⟧ \vDash_L ⟦P⟧$ and $⟦S⟧ \vdash_L ⟦P⟧$

(437) *Meaning of S*
The meaning of a sentence S is the conjunction of its entailments derivable from its Logical Form

These definitions have several edges. Firstly, they relate logical form to linguistic empiricism. Entailments are those aspects of sentential meaning on which language users converge. Entailments, therefore, are sentences, not formulas. With his syllogisms, Aristotle did not discover new cognition – he just made explicit the oldest insights of talking heads. Human beings may disagree on the meanings of words, but entailment is a decidable property among linguistic laymen, as is claimed correctly by Chierchia and McConnell-Ginet (2000).
Secondly, the statements define a lower bound to logical form. It invokes formal reasoning to connect entailments to logical form, by assuming that some entailments are so elementary that they do not have any other entailments themselves. Logical form must be so rich that formal deduction of entailments is possible, according to some logic.
Thirdly, we can use (436) and (437) to define an upper bound to the notion of entailment. Entailments are those propositions that are connected to sentences by judgements of language users. We assume that language users only have (linguistic) judgements about propositions that are conceptually – say, lexically – related to the base. Few language users are likely to acknowledge entailments outside the lexical realm of the base – only the trained logicians among them are in danger of doing so. We assume that all the material in entail-

ments of a sentence is delivered by the lexical components of the sentence. Every entailment is built completely from semantic material available in the sentence. Because of this – quite realistic – restriction, we can neglect entailments produced by purely logical combinatorics, like the logical equivalence of *p* and *p & p, p → p, p & (p ∨ ¬p)* and so on. We define an entailment of A as the shortest representative of its logical equivalence class within A's lexical realm.

A sentence entails all of its entailments simultaneously. As a matter of fact, definition (436) gives rise to the following conjecture:

(438) If P_1, ... P_n are all entailments of S, the logical form ⟦S⟧ is equivalent to the conjunction ⟦P_1⟧ & .... & ⟦P_n⟧

Since entailments may be complex propositions themselves, the explicit conjunction of all entailments will be redundant. Still, the shorter and more economical representation is basically conjunctive. That is, if every ⟦P_i⟧ itself is spelled out as a conjunction p_i & ... & p_n, the representation ⟦S⟧ is the conjunction of all those propositions p_j that occur in the representation of at least one P_i. The logical form, then, is a kind of conjunctive union of all entailments. The idea that logical form is basically conjunctive is advocated by Pietroski (2006, 2011) from a purely linguistic point of view, and by Reckman (2009) from a computational-linguistic point of view. It is advocated and integrated in the DELILAH semantics in chapter 2 of this monograph as Flat Logical Form.

For the logical form to be the source of all entailments, it combines semantic contributions of phrases by applying one meaning to another, bringing one semantic contribution under the logical influence of another. In particular, semantic contributions can be excluded from inference by wrapping them into intensional domains, or be made referential dependent upon other semantic terms. Even if a phrase is a proper part of a sentence, its semantic contribution to the sentence might be non-existent, mutilated or only partial. In exactly this sense, semantic combinatorics is *multiplicative* and *selective*. It behaves like intersection, in accumulating interpretative conditions on the individual terms. From the consideration above and the conjunctive nature of the logical form, it follows immediately that the notion of *semantic consequence* ≺ induces an antisymmetric and transitive ordering between the logical form and any entailments of the sentence; in the definition, ". ≺ ." abbreviates the conjunction of ". ⊨ ." and ". ⊢.".

(439) If ⟦S⟧ is the logical form of S and S entails P, then ⟦S⟧ ≺ ⟦P⟧

In this ordering, the whole – the conjunction of entailments – is ordered 'below' the proper part – the individual entailment. Its antisymmetry follows from the restriction mentioned above, that entailments are built from lexical material in the sentence and that they are the 'shortest' representatives of their equivalence class.
With respect to the ordering, it is of some importance to note that the notion of semantic consequence is not conservative in the sense in which unification is (cf. (423)). In particular, extension of the logical form of a sentence does not preserve semantic consequences the way extension of the syntactical form preserves constituency.

(440) *Non-preservation of semantic consequences*
If ⟦S⟧ is the logical form of S, S entails P and ⟦S⟧ ⊰ ⟦P⟧, there are embeddings S ⊔ E of S such that ⟦S ⊔ E⟧ ⊰ ⟦P⟧ is not valid.
If $\mathbf{P}_S$ is {P|⟦S⟧ ⊰ ⟦P⟧} and $\mathbf{Q}_E$ is {Q|⟦E⟧ ⊰ ⟦Q⟧}, then $\mathbf{R}_{S \sqcup E} \subseteq (\mathbf{P}_S \cup \mathbf{Q}_E)$

Among these syntactically conservative embeddings that do not preserve semantic consequences we place all those phrases which are called *plugs* in the literature on presuppositions and their projections. A ready example is a propositional frame introduced by an intensional verb, taking a proposition as its complements. Embedding a proposition under an intensional verb may not preserve its presuppositions, which would have been entailed otherwise. Even more transparent is the operation of disjoining sentences: whatever is entailed by S is not entailed by *S or P* unless it is also entailed by P. For this reason – a conservative embedding of the whole does not preserve entailments – the ordering imposed by the notion of entailment is the opposite of the ordering imposed by constituency.

The conjunction of entailments of a sentence is not projected from its syntactic structure. Nothing in the syntax gives rise to flat representations. What is more, not every constituent, not even every constituent at a certain level of derivation, gives rise to any particular entailment. An argument of a verb, for example, does not induce any entailments of its own, although it is generally the prototypical constituent. Quite the opposite is true. Entailments – testable entailments – are almost exclusively produced by combining the semantic contributions of *several* constituents. In order to produce entailments, the semantic contributions of constituents apply to each other. The agenda for these applications is dictated by the constituent structure – the syntax – but the outcome is not dictated by syntax. The result of applying two semantic contributions to each other is solely defined by their internal structure, since the application itself – mostly modeled by lambda conversion – is too gen-

eral to calibrate entailments. Since Montague (1972), we have been taking the lambda terms themselves as introducing enough structure to guarantee a structured outcome.

Therefore, we must assume that some non-conservative but very restrictive process of applying semantic contributions is active in order to produce the conjunction of entailments. This process is *functional composition*.

(441) If f_i is the lambda term associated with constituent c_i of S, i.e. $f_i = [\![c_i]\!]$, there is some protocol according to which the composition $\prod_{i=1..n} f_i$ or $(\ldots\ (f_1 \circ f_2) \circ \ \ldots \circ f_n)$ is arranged such that the result of applying that protocol to the composition is the logical form $[\![S]\!]$.

The protocol that is assumed accounts for both the syntactical structure and the invariants of the underspecified semantic representation. Being sensitive to syntactical structure as well as the underspecified semantic representation, the protocol embodies the compositionality of interpretation. Since constituents and entailments do not match, the protocol accounting for the relationship between the two is neither a syntactic nor a semantic computation, and it is not restrictive at all. Such a protocol was alluded to in chapter 2, taking care of the transformation of Stored Logical Form into Applied Logical Form and Flat Logical form. Note however that the conjunctive result of the composition is not caused by the protocol, but to the internal structure of the applied terms themselves. The result of the composition is a product, with all kinds of terms restricting each other. The nature of this composition is best understood by considering that an entailment cannot be traced back to a constituent and that all entailments are simultaneously entailed by the whole.

4.4 THE DENIAL OF STRUCTURE

In the previous section, interpretation was defined as a procedure to extract the necessarily underspecified semantic representation from the unified syntactical structure, to identify the contributions per constituent and to apply these contributions to each other. The result must be a conjunction of propositions – the entailments. This leads to the following overall characterization of the 'syntax-semantics interface.' The interpretation of an additive constituent structure implies the multiplication of the interpretation of the constitu-

ents, by the protocol referred to in (441); the latter 'spells out' a conjunction of propositions, by pure

(442) *Interpretation*
$[\![(...(A_1 \sqcup A_2) \sqcup ..... \sqcup A_n)]\!] \Rightarrow (...([\![A_1]\!] \sqcap [\![A_2]\!]) .... \sqcap [\![A_n]\!]) \Leftrightarrow p_1 \,\&\, ... \,\&\, p_k$

It shows that the interpretation inverts the additive algebraic structure of syntactical form – the additive, conservative, highly structured unification of complex signs – into the multiplicative logical form – an invasive, flat chaining of small clauses, conjoining to fully-fledged entailments.
The directionality of the interpretation process does not mean that in grammar the inverse process does not exist. In fact, we identified the process of generating sentences from independent meanings as a form of 'backpropagation' of this kind; the operation is referred to by the sign $\langle\!\langle . \rangle\!\rangle$.

(443) *Generation*
$\langle\!\langle\, p_1 \,\&\, ... \,\&\, p_k \,\rangle\!\rangle \Rightarrow \langle\!\langle (...([\![A_1]\!] \sqcap [\![A_2]\!]) .... \sqcap [\![A_n]\!]) \,\rangle\!\rangle \Rightarrow$
$(...(\langle\!\langle A_n \rangle\!\rangle \sqcup \langle\!\langle A_2 \rangle\!\rangle) ... \sqcup \langle\!\langle A_1 \rangle\!\rangle)$

Given some simultaneous semantic constraints, generation retrieves some composition of meanings compatible with these constraints and a unification of the associated complex symbols. This picture is misleading, however, in that functional composition is not procedurally 'in-between' the conjunction of entailments and unificational structure. Rather, the composition arises from unification, and is supposed to convert to the original conjunction. Generation is about finding out whether there is some composition to provide the original, structure-independent conjunction.

(442) and (443) define the grammatical cycle from form to meaning and back as the consecutive application of the operations $[\![.]\!]$ and $\langle\!\langle . \rangle\!\rangle$. The first operation assigns meanings to forms; the second assigns forms to meanings. Their composition is antimorphic: it maps addition onto multiplication and *vice versa*. That is what negation does in the calculus of propositions, according to the De Morgan laws, or what complementation does on sets.

(444) $\neg(p \vee q) \quad \leftrightarrow \quad \neg p \,\&\, \neg q$
$-(A \cup B) \quad = \quad -A \cap -B$

Negation and complementation are both antimultiplicative and antiadditive. Neither syntax nor semantics, however, are complete Boolean algebras, hosting both upward and downward operations and complementation. At best,

syntax and semantics are monoids, generated by one operation. Syntax is defined by a conservative, addition-type operation (say: concatenation-by-unification), for which complementation is not defined. Semantics is defined by an invasive, multiplication-type operation – function composition – for which complementation and identity are defined. Interpretation is some mapping from one additive monoid onto another multiplicative monoid. Thus, interpretation, like negation and complementation, is antiadditive. That is why we can say that semantics accounting for inference denies unification structure, while observing compositionality. In the same vein, generation is antimultiplicative, by letting structure negate meaning. Because generation is even less understood than interpretation, for the rest of this chapter we will stick with interpretation.

Let us refer to (442) as the *antiadditivity of full interpretation*. The antimorph nature of full interpretation has consequences for various approaches to natural-language grammar and natural-language processing. The main consequence is this. The concepts with which we structure syntax are quintessentially different from the concepts that buy us semantic inference. There is no monotony when going from unification to entailments, or the other way round: from meanings to sentences. This affects shallow processing, for example. Shallow processing with ambitions beyond tagging phrases lives on the idea that somehow aspects of meaning arise from counting and recounting and modeling co-occurrences. The means by which we can make educated guesses on the form of sentences up to the identification of referential objects, however, cannot be monotonously extended to cover entailments. They serve different algebras. The antimorphism of full interpretation also affects those grammatical theories that appeal to *deep* syntactical operations for meaning to come about, like the 'classical' theory of logical form in generative grammar.

4.5 THE MISMATCH OF STRUCTURE AND MEANING

The idea that the meaning of a sentence is determined by its construction is generally attributed to Frege, but wrongly so, according to Janssen (1983). Yet, as a mathematician, Frege must have been well aware that the reference of a complex phrase somehow lives on the way in which its parts contribute to that reference, and that this contribution is delivered by syntax. For it makes little sense to assume that the referential potential of a sentence is not somehow

dependent on its construction. This does not imply, however, that the building blocks of syntactical construal match with the jewels of semantics. As a matter of fact, it is not difficult to show that in many constructions entailments and constituents diverge seriously. A main source of incongruence is type incongruence: entailments are propositions, but most constituents do not project propositional meanings. Take, for example, DPs. Most DPs have meanings that are only remotely related to propositions. Nominalizations, however, may introduce propositions, which can even be entailed. Below, three examples of embedded nominalizations in the relatively independent subject position are given, with an infinite scheme for a proposition projected by it.

(445) The destruction of the city by the enemy is solicited by our own army.
(446) The king welcomed the destruction of the city by the enemy as a blessing.
(447) The destruction of the city by the enemy was finally avoided by a simple trick.
(448) The enemy destruct-X the city.

For two reasons, a possible entailment based on the nominalization is not independent. Firstly, all virtual entailments based on the nominalized subject need tense, which can only be derived from scrutinizing the tense and the aspect of the matrix. And, secondly, if tense and aspect can be borrowed for some enemy-destroying-city-type of proposition, the veridicality profile of the matrix completely determines whether this proposition – or even its denial – is factually entailed. Therefore, not even a propositionally inclined DP projects 'its' entailments autonomously.

In this section, we will consider three constructions in which the mismatch of constituents and entailments is evident, although sentences and propositions seem to be available. In exception phrases, entailments pop up which almost look opaque if one focuses on the lexical material. In comparatives, entailments are suppressed that seem almost inevitable. In ellipsis, entailments are evident without constituency available to carry that load. It is not accidental that these sample constructions are all related to coordination. Recall that entailments are somehow – covered – coordinates of logical form. When the syntactical configuration gives rise to its own explicit coordination, the clash between form and meaning can hardly be overlooked. In Cremers (1993a), it was argued that the procedures for reconstructing free explicit coordination are in fact meta-grammatical (see also section 1.7.4).

4.5.1 The mismatch in exception phrases

In exception phrases, the referent of a constituent that is licensed by an *exception*-like coordinator is somehow set apart with respect to a predication. They come in two modes. Either the constituent is excluded from predication over a class, or the constituent is included in the predication. Here are Dutch examples of the first (exclusive) and the second (inclusive) mode, respectively.

(449) Op een historische roman na heb ik Marie haar boeken allemaal gelezen
'except for a historical novel have I Marie her books all read'
I read all Mary's books with the exception of a historical novel.
(450) Henk heeft behalve met Dylan (ook) veel opgetreden met The Boss
'Henk has except with Dylan (also) much performed with The Boss'
Apart from Dylan, Henk (also) performed a lot with The Boss

Exception phrases have some intriguing syntactical-semantic details. The set of items that can introduce the excepted constituent is limited. Often, they look like prepositions, but semantically, they behave like coordinators. In particular, they are sensitive to the polarity of one constituent in the main clause, which acts as a counterpart to the constituent that is excepted. If this constituent is upward entailing with respect to its internal domain, the exception phrase is interpreted inclusively. Otherwise – for example when the counterpart is a universally quantified phrase – the exception phrase is interpreted exclusively. In both modes, the exception phrases can occur in almost all positions in which adverbial adjuncts can occur, without change of meaning, though with possible consequences for information structure. Moreover, the complement to the excepting operator can be an elliptical phrase, which complicates the analysis considerably. Both the positional freedom and the elliptical option are demonstrated below.

(451) Ik heb geen filosoof alcohol durven schenken, behalve Arie wijn.
'I have no philosopher alcohol dare present, except Arie wine'
I did not dare to present alcohol to any philosopher, except for wine to Arie
(452) Ik heb behalve Arie wijn geen filosoof alcohol durven schenken.
(453) Behalve Arie wijn heb ik geen filosoof alcohol durven schenken.
(454) Ik heb, behalve Arie wijn, geen filosoof alcohol durven schenken.

To appreciate the mismatch between constituency and entailments, we focus on a simple exception sentence in an exclusive mode, assuming some bracketing which indicates syntactic form while abstracting from the labelling. The structured alternatives are listed in (455) - (456), with the except phrase occurring post-verbally, pre-verbally and adnominally, respectively.

(455) Alle filosofen waren dronken behalve Socrates.
All philosophers were drunk except for Socrates.
(456) [[[Alle filosofen] [waren dronken]] [behalve Socrates]]
(457) [[Behalve Socrates] [waren [alle filosofen] [dronken]]]
(458) [[[Alle filosofen] [behalve Socrates]] [waren dronken]]

Each of the syntactical alternatives has three prominent entailments, presented here in Dutch.

(459) [[Niet alle filosofen] [waren dronken]]
Not all philosophers were drunk
(460) [Socrates [was [een filosoof]]
Socrates was a philospher
(461) [Socrates [was [niet] dronken]]
Socrates was not drunk

In former days, one could have maintained that for the simple reason that three differently structured sentences share their semantics, the three sentences must have a common syntactical ground. In modern syntax, this reduction would involve some operations like *extraction-from*-DP or *insertion-into*-DP, which hardly have an independent motivation and require a serious complication of the combinatorial equipment, beyond control. In any case, the common derivation would be inspired by semantic similarities, not by syntactic processes – there are no other structures in which a proper part of a DP (as in (458)) ends up or begins as a leftmost specifier (as in (457)), without semantic consequences.

Since it is unlikely, therefore, that in serious syntax the three options are in each other's derivations, we had better look at the projection of the syntax on the entailments. Note, firstly, that none of the entailments is entailed by the matrix sentence (without the exception phrase) clause alone. The constituent *Alle filosofen waren dronken*, though a proposition as such, does not give rise to any independent entailment. This already shows that being a constituent, or even being a constituent of the right type at top level, is not a sufficient condition for providing entailments.

Constituency is not even a necessary condition for a phrase to provide, or to contribute to, an entailment. Scrutinizing the major parts of the entailments, it is evident that none of the following phrases quintessentially occurring in the entailments can be derived syntactically from proper constituents of the original, for example by reversed unification or decomposition. The following constituents of the entailments are syntactically untraceable.

(462) Niet alle filosofen *not all philosophers*
was niet dronken *was not drunk*
was een filosoof *was a philosopher*
niet *not*

Remarkably, *none* of the immediate entailments can be constructed as an amalgam of proper constituents of one of the entailing sentences. As a consequence, we can safely state that there is no transparent relation between constituents of an exception sentence and its prominent, non-trivial entailments.

4.5.2 The mismatch in comparatives

Comparatives are common constructions among the languages of the world. Though the syntactical variety is huge and subtle at the same time, the characteristic overall pattern can be captured by a configuration in which something is predicated over something with a certain degree that is implicitly compared to the degree to which the predicate applies to something else. It is a common feature of comparatives that there is only one explicit grade phrase in the sentence which is in a construction with some comparison particle. The second grade phrase is predominantly absent – a phenomenon known as subdeletion. The interpretation of these constructions, however, involves three propositions: the original graded predication, the secondary graded predication and the proposition containing the comparison (Von Stechow 1984). Below you will find the syntactical and semantic schemes of a simple Dutch example:

(463) [[... grade1...predication1] [comparison phrase ... predication2]]
[[Jan wandelt sneller] [dan Kees fietst]]
Jan walks faster than Kees bikes

(464) ⟦grade1 predication⟧ and ⟦grade2 predication⟧ and ⟦comparsion of grades⟧ ≈
⟦John walks at a certain level of speed⟧ and
⟦Kees bikes at a certain level of speed⟧ and
⟦the level of speed at which John walks is higher than the level of speed at which Kees bikes⟧

The most intriguing aspect of comparatives is that neither of the two sentences – elliptic or full – involved is entailed as it stands. The first sentence contains an expression of grade – in the example above, a morpheme – that is not part of the proposition induced as an entailment by that first sentence. The sentence does not entail that Jan walks fast. The second sentence does not contain a grade expression but it is not entailed either. The third proposi-

tion is a derivative of the relation that the comparison phrase imposes on the degrees to its left and its right – present or not – but is as such not projected by any major constituent of the original sentence. The fact that three different propositions are entailed by the construction and that these entailments are partly induced by sentential-looking proper parts of the constructions underlines the analysis of comparatives as coordination, rather than subordination, by Hendriks (1995) and others.
For some comparative constructions, it is even difficult to maintain that they have an underlying clausal structure.

(465) You are better than no one (and no one is better than you)
(from Bob Dylan's *To Ramona*)

This line is meaningful – expressing that everybody is at least as good you are – but no clausal extension is.

(466) # You are better than no one is / has ever been / could ever be

The meaningful occurrence of the negative quantifier complies with Hoeksema's (1983) observation that non-clausal comparatives do not license negative polarity items, where clausal comparatives do. Notwithstanding its non-clausal syntax, its interpretation still involves a conjunction of comparison clauses.

(467) ⟦You are better than no one⟧ = ⟦you are good to some degree⟧ and ⟦everybody is good to at least that degree⟧

Clearly, the second clause cannot be derived by syntax from the *than* phrase.

What we observe here is that there is some evident underlying sentential structure in the construction, but that these sentences quintessentially deviate from entailments in syntactically almost inaccessible but semantically prominent ways. The deviation is even more prominent in simple adjectival predication. Sentence (468) entails neither that Jan is smart nor that Marie is rich.

(468) [[Jan is slimmer] dan [Marie rijk is]]
'Jan is smarter than Marie rich is'
Jan is smarter than Marie is rich

The incongruence between sentential structure and entailments – propositional by nature – is not an accidental feature of comparatives in Germanic languages. It is widespread, and it is even a defining property of the construc-

tion in the languages of the world. Here too, syntax almost misleads semantics, and semantics contains few clues as to syntactical structure.

4.5.3 The mismatch in ellipsis

Gapping is the prototypical source of incomplete sentences, interpretable only in heavily conditioned, language-dependent verbal contexts. The verbal context is typically provided by coordination-like ordering. If these conditions are met, the interpretation is perfect. Here are some typical examples of gapped structures, indicated by italic type.

(469) Ik weet dat de jongens geen boeken willen en *de meisjes geen films*
'I know that the boys no books want and the girls no movies'
I know that the boys don't want books and the girls don't want movies

(470) Komt Henk vandaag met de trein? Nee, *morgen en met de fiets*
'Comes Henk today with the train? No, tomorrow and with the bike'
Does Henk come by train today? No, tomorrow, and by bike

(471) Niemand heeft voor Aruba gekozen, en *bijna iedereen voor Bonaire*
'Nobody has for Aruba chosen, and almost everyone for Bonaire'
Nobody choose Aruba, and nearly everyone Bonaire

In the study of grammar, numerous analyses have been proposed to ensure that the non-sentential proposition could be reconstructed. In Neijt (1980), gapping was even hailed as a specimen of core syntax. Thereafter, ellipsis was completely ignored by grammarians, or treatments were proposed with *ad hoc* mechanisms, culminating in a dedicated licensing construction in Merchant (2001). The interpretative approach of Dalrymple, Pereira and Shieber (1991) goes beyond syntax in order to stress interpretability. The problem with syntactical reconstruction is that the missing parts of the ellipsis are not available as a constituent – this was the basic observation that made Neijt (1980) move from reconstructing elided material to defining the relationship between remnants. The real problem is that normal syntax cannot provide constituency because anything can be a remnant, in particular parts that are not or even cannot be a target for reconstruction rules, like non-finite verbs.

(472) Kees is naar Amsterdam gefietst, en Jan gelopen.
'Kees is to Amsterdam biked-PART and Jan walked-PART.
Kees biked to Amsterdam and Jan walked.

In sentences like these, any effort to produce a structure that can provide the interpretation for the second sentence by general syntax is in vain. Such an

effort is bound to call upon dirty operations on non-constituents, as there are not enough moves available to save the day. Yet, language users do not have serious problems in interpreting ellipsis. And because explicit and implicit conjunction is a very generous source of ellipsis, language users do not have problems computing entailment. Every speaker of Dutch will recognize that (472) entails both propositions in

(473) Kees is naar Amsterdam gefietst.
(474) Jan is naar Amsterdam gelopen.

The second entailment clearly stems from the elliptical phrase. It is equally clear, however, that the entailed proposition is not provided by unificational syntax.

4.5.4 A conjecture on the interface

An open-minded look at the four types of examples of the separation of constituency and entailment and numerous other mismatches leads to the idea that this separation is neither incidental nor accidental. We might conjecture that, except for some full sentences, constituents do not project any entailment, and that entailments cannot be traced back to a non-sentential constituent. This conjecture is made explicit below.

(475) *Dissociation conjecture*
If A is a proper constituent of S and S entails P then ⟦A⟧ does not entail P and P is not equivalent to ⟦A⟧.

With a proper constituent we mean every constituent that is not a full sentence, implicitly or explicitly coordinated with other full sentences. The conjecture excludes any naive mapping of structure on meaning, of syntax on semantics and of unification on composition. It says that ultimately decent grammar does not enjoy compositionality.
According to (475), meaning does not come for free with quantifier movement and the like. Syntactical operations may get us somewhere, but not to the semantic finish. For deep interpretation, grammar has to get rid of structure. According to (475), the *syntax-semantics interface* calls for methods beyond merge, unification, feature checking, and the like. These methods may be type-0 in the Chomsky hierarchy, and not subject to elegant principles of locality, economy, type-shifting, binding or whatever. They may appeal to that part of cognition that is beyond strong artificial intelligence, to pick up on

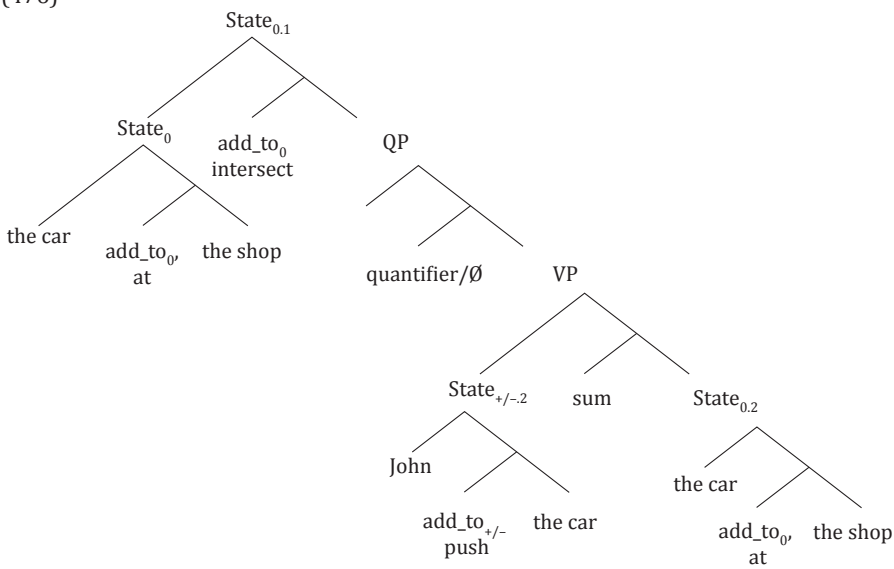
Roger Penrose's (1989) lead: they may be computable, but this computation does not necessarily reflect established principles of grammar. In that sense, (475) conjectures that syntax and semantics are not functionally related but, rather, they are complementary branches of linguistic analysis. To get to meaning, you have to go beyond unification. To compute entailments, forget about constituency.

4.6 THE LEXICON AS AN ORACLE: THE CASE OF *BEHALVE*

In the preceding sections, and even in all the preceding chapters, it was argued that syntax and semantics, unification and composition cannot carry the full load of deriving linguistic truisms. Again, we will invoke the lexicon to provide the bits and pieces of the final theorems – those truisms that explicitly connect texts by dubbing one text an entailment of the other. In addition, we need algorithms which live neither on unspecific unification nor on nervous, typed composition to assemble the bits and pieces offered by the lexicon as input into the final act of constructing meaning as the simultaneous presence of relevant entailments.

The idea that the lexicon stores specific linguistic knowledge is quite common among computational and other linguists. This view can even be called traditional: the lexicon/constructicon contains all information that cannot be derived by other means or is unpredictable. The idea that the lexicon feeds detailed semantic structure into the derivation can be traced back to Montague (1972), the protagonist of event semantics for verbs, and – for broad-minded lovers of history – to Russell's theory of definite descriptions. The essence of storing rich, very rich semantic structure in the lexicon is that it does not interfere with type-logical complexity – the semantic contribution of a phrase is type-driven but not type-structured. A verb, for example, can come with extensive aspectual and pre-suppositional structure, but it may also introduce a simple property. An achievement will bring with it more semantic material than a simple event verb (see, *e.g.* Arsenijevic 2006). Below you will find the semantic representation that Arsenijevic (2006) claims for sentences with perfective telic verbs, like *John has pushed the cart to the shop.* It involves at least three states, connected by relations between propositions.

(476)



Let us now focus on a few of those elements that, like no other, influence the inferential structure of a sentence: the functional heads. Montague (1972) awoke linguistics by assigning (477) to the simple determiner *the.*

(477) $\lambda_{\wp_{<ssett>}} \lambda\Re_{<ssett>} \forall x_{<se>} [\wp\{x\} \Rightarrow \Re\{x\}]$.

The determiner was stored as a function of type <<ssett, ssett>, t>, that is, a relationship between properties-in-intension, introducing a partial ordering – as an implication or subsumption of sets – in the semantic structure of any sentence which it was part of.

In the same vein, we might ask what a functional head like *behalve* 'except' must look like in order to ensure the entailment pattern for exclusive readings that was established in section 4.5.1. It is repeated here.

(478) Behalve Socrates waren alle filosofen dronken
Except for Socrates, all philosophers were drunk

entails

Socrates was een filosoof — *Socrates was a philosopher*

and

Niet alle filosofen waren dronken — *Not all philosophers were drunk*

and

Socrates was niet dronken — *Socrates was not drunk*

All of these entailments are induced by the exception phrase or, better, by its head *behalve*. None of the entailments, however, can be traced back to any single constituent. Therefore, we must rely on *behalve*'s lexical specification for providing the entailment pattern: if the constituent structure cannot identify entailments, the functions-to-be-composed must do so. The phrasal semantics is the only input to the identification of entailments when syntax and unification cannot deliver. The three entailments of exclusive *behalve* are tantamount to conjuncts at the high-level logical form of a sentence governed by *behalve*.
In order to specify the entailment pattern as a part of the lexical semantics, we have to determine some essential anchors in the sentential structure. First, it must be noted that the complement of *behalve* needs some co-typed counterpart in the matrix. This counterpart must be downward entailing with respect to its internal domain for *behalve* to get an exclusive reading. Moreover, the internal domain itself is needed for its contribution to a predicative statement about the complement of the exception phrase. Also, the negation of the main clause's quantificational structure must be constructable, and this again may depend on the semantics of the counterpart phrase. Secondly, temporal and aspectual characteristics of the main clause must be isolated and targeted in order to get the aspects of the entailments right. Thirdly, the semantics of *behalve* must guarantee the simultaneous validity of at least the three entailments; they are in conjunction with each other. Summarizing, the semantics of exclusive *behalve* in the constructicon for Dutch must informally be as follows.

(479) *Behalve* denotes that function ***f*** from a proposition P and an object B to propositions such that
if P is equivalent to R(S), R is of type of B and equivalent to Q(N) with Q downward entailing in N,
then ***f***(P, B) is the conjunction of
(i) the denial of P
(ii) the denial of the application of B to S
(iii) the assertion that B is N according to the tense of P

The definition assumes that in the main sentence the semantics of the counterpart to *behalve*'s complement is detectable. Its detection, however, will require some non-unificational algorithm, although the detection must be syntactically valid: the counterpart is supposed to be a proper constituent of the main clause to which the *behalve* phrase has been attached. The detection, therefore, requires some post-derivational inspection of the structure. Seen in this way, part of the semantics of *behalve* is a dedicated algorithm to determine the target constituent. In the formulation above, this algorithm

– too dirty to spell out here – is absorbed in the *if*-clause. It is as double-layered as Pythian talk.
More formally, the semantics of exclusive *behalve* can be stated this way:

(480) $f_{\text{behalve-exc}}(Q_{\beta\alpha}(N_{\beta})(S)_{\alpha\tau}, B_{\alpha}) = B(N) \,\&\, \textit{not}(Q(N)(S)) \,\&\, \textit{not}(B(S))$

When fully applied, the composition of this function with relevant meanings in its environment yields a conjunction of logical forms. One may hope that a generator can turn these logical forms into decent Dutch, or decent Swahili, for that matter. If so, the constructicon helps to overcome the predictable and built-in limits of good grammar.

4.7 THE INCOMPLETENESS OF GRAMMAR

Gödel's Incompleteness Theorem is both a fairy tale and rocket science. It repositions human intellectual analysis and understanding. It says that we are able to understand some very complex processes, without being able to fully compile them, to abstract from them completely. In his criticism of claims amounting to the perspective of creating extra-human supra-intelligence, Penrose (1989) over and over refers to Gödel's magnificent result, and with good reason: a system cannot prove its consistency, even if we know that it is consistent. And what is best: Gödel *proved* this to be the case – at least he convinced human intellect.

This chapter is not about turning grammar into mathematics. There is always that quotation from Daniel Kehlmann's *Die Vermessung der Welt* "Measuring the World" according to which Gauss tells Wilhelm von Humboldt, who claims to be a researcher too, that linguists long for mathematics but are not smart enough and deal with home-cooked poor logic. Still, grammar running on computers is a formal system of the type that is subject to Gödel's Incompleteness Theorem. It is, however, dealing with more complicated, multi-layered and more intensional objects than numbers. It is crucial to Gödel's Incompleteness that the class of theorems beyond proof of rejection is well-defined, and intrinsically restricted. Gödel actually proved that in arithmetic a two-place predicate *X is a proof of Y* introduces inconsistency. He proved that for a system to be consistent certain theorems are bound to be beyond computation. He proved that the set of arithmetically provable sentences

is a proper subset of the set of arithmetic truisms, and that the difference between these is well-defined.

For computational grammar, that is, for grammar that operates by computation unsupervised, certain theorems conflict with the internal mechanisms of grammar. Certain theorems are incompatible with doing syntax syntactically and semantics semantically. Our suggestion is that these theorems are exactly those that connect the yield of syntax to the yield of semantics. Let us write [S] for the syntactical analysis or the constituent structure of a sentence S, according to everything that syntax buys us. Let us assume that we have a decent calculus of entailments – we do not have such a calculus yet, but there is no reason why we could not find one – and let us simply call the relevant relation *entails*. Then we can state the incompleteness of grammar as the incompatibility of a statement about entailment between structured sentences with both unification and composition.

(481) *Incompleteness of grammar*
For [S] and [P] being consistent analyses of S and P respectively, and if *S entails P* means that their logical forms are related by $[\![S]\!] \prec [\![P]\!]$,
the proposition *[S] entails [P]* can neither be proven nor disproven by unification or composition.

The idea is that syntactical means – based on unification – are insufficient and inadequate to identify [P] as an entailment, and that semantic means – based on composition – are insufficient and inadequate to reconstruct [S] as an entailer of [P]. From a processing point of view, syntax is not strong enough to bring us to entailments, and composition is too invasive to bring us back to form. Together, they do not suffice to span the grammatical cycle mechanically. Since a homomorphism between unification and composition cannot be constructed, an attempt to prove theorems of the form *[S] entails [P]* is bound to fail. This failure identifies the class of theorems as filling the gap between grammatical statements that can be deduced and linguistic truisms that cannot. S may entail P, but *[S] entails [P]* cannot be proven by grammatical means. It is essential to acknowledge that incompleteness (481) is not a defect of grammar, but rather an asset. It qualifies the art of interpreting natural language in one of those domains that can be formalized without expanding to a totalitarian, holistic philosophy of everything. Though language is everything, not everything is language, Van Benthem claimed, and he is right. Even the grammar of natural language cannot justify itself. It is not the last word on how it is. The grammarian and her constructicon will always be needed for the final touch.

4.8 THE FRUIT OF INCOMPLETENESS

Several strategies are known for repairing the non-monotony of syntax and semantics and, thus, for avoiding incompleteness. Here are four ways to free your favourite grammar from incompleteness.

(482) (1) stick with underspecified semantics
(2) abstract from logical form
(3) abstract from constituent structure
(4) abstract from grammar

The first strategy is practised in HPSG and Head-Driven Construction Grammar (Boas and Sag 2012) and part of a well-defined computational model. The second strategy is shared by Generative Grammarians and Construction Grammarians; it is not confessed, but just practiced. Montagovianists and Lambek-style Categorial Grammar coined the third strategy structural completeness. The fourth strategy is common in Bayesian processing, as a survival strategy for robust parsing.

One way of implementing the third strategy is enriching syntactical derivation, up to mimicking lambda conversion in the syntax. This was practiced by Montague (1972), and – on a more principled basis – by many categorial grammarians after him (Moortgat 1988, Carpenter 1997, Morrill 1994 and 2011, Hendriks 1993). Under this strategy, syntactical derivation is made responsible for scoping effects and other forms of semantic dependency. It leads to the blurring of syntactical structure and derivation, based on the structural completeness of certain categorial calculi: if a sentence can be proven to be interpretable, it is so under any bracketing or grouping of constituents. By structural completeness, many derivational options are created that have no effect at all on constituency or linear order. The result is spurious ambiguity, which has to be fought afterwards (Dowty 1988). Yet, the distribution of cases and roles in *Every man loves some women* is not affected by which women are loved by which man in a model for that sentence. The strategy to flexibilize the derivation makes syntactical structure obsolete. It relinquishes the beautiful and intriguing linguistic generalization that syntactical structure is essentially independent of denotation – the autonomy of syntax, a major fact about natural language as a cognitive system. P and notP do not differ as fundamentally in syntax as they do in semantics.

The other real strategy to deal with the antimorphic tension between structure and interpretation is to be satisfied with underspecified semantics: that level of semantic specificity that can be reached by pure unification. This is the dominant attitude among deep processing computationalists (Copestake *et al.* 2005, Bunt 2008, Egg 2010), and maybe among logical-form theoreticians. It refrains from invading the syntax, and it keeps the grammar as a clean and computable as possible. Like Egg (2010), Boas and Sag (2012:95) even sees advantages to it:

> *"Scope underspecification is the heart and soul of M(inimal) R(ecursion) S(emantics), which was originally developed for computational purposes. Scope resolution ... is a difficult and currently unsolved research problem. Hence, having semantic representations that do not require full scope resolution is a significant advantage in much computational work, e.g. machine translation. However, ..., it may be useful in psycholinguistics, as well."*

Underspecified semantics avoids incompleteness, but does not identify linguistic meaning as an independent object in its own right, living in a realm in which syntax is no longer needed. It adheres to a representation that is meaningful only in relation to a syntax-semantics interface. Underspecified semantics does not present a logic, with computable operations on meaning. Underspecification implies raising the question whether meaning exists.

In generative grammar and construction grammar, the nature of logical *form* is not a big thing. Croft (2001:212), for example, presents the semantic scheme for a Tzotzil sentence as a graphic map of the annotated sentence on some predication-like sequence, without accounting for the details of the mapping. Construction grammarians tend to represent aspects of meaning in graphics, giving little attention to compositional or formal matters. With the possible exceptions of highly theoretical approaches like those of Kracht (2004) and Stabler (1992), generative grammarians only rarely pay tribute to the precise derivation of generalized logical form. It is unlikely, however, that generative grammarians would consider logical form trivial. Broekhuis and Dekkers (2000) suggest that in a grammar model with split derivation for phonological and logical form, the former but not the latter may be subject to optimalization, thereby maintaining the 'classic' set of parameters and principles for logical form. In neither grammatical tradition are the semantic representations chosen (pictures, graphs, formulas) explicitly linked to constituent structure. Logical form is mostly anecdotic, derived stepwise. Of course, incompleteness is avoided this way, but logical form as a product of grammar

is not achieved. Again, independent identification of a semantic object is not sought after. Quite the contrary, grammarians of generative and constructional persuasions give the impression that they do not consider linguistic meaning as an independent object.

In Bayesian approaches to natural language processing, it is becoming more and more accepted to claim that grammar as such can be circumvented (*e.g.* Frank, Bod and Christiansen 2012). In this realm, grammar is at best a necessary evil, and particular grammatical specifications as such are hardly a topic of debate. The journals of computational linguistics are mostly about processing on the basis of some given annotation, and not about which grammatical annotation best serves computational goals. As a consequence, the internal organization of the grammar is seen as external to processing. Incompleteness in the sense of (481) stays out of sight.

It pays to see the struggle for computing towards and from meaning as the price for incompleteness. Paying this price frees syntax from semantic overload, and frees semantics from contingent syntactical bookkeeping. It prevents us from computing distinct syntactical derivations to accommodate purely semantic diversity. It makes semantics transparent, as it does not have to reflect syntactical accidents. Syntax brings meaning to a certain point – Stored Logical Form, as we called it in chapter 2. After that point, syntax is only *exploited* by meta-algorithms to compute fully specified meaning, but syntax is no longer in charge. In this sense, the overall computational properties of grammar are beyond the properties of syntax. The meaning of a sentence is Cognition with a capital. The fact that human beings are able to act as if sentences are meaningful marks an astonishing convergence of biological and social evolution. Yet, the fact that we dynamically assign meanings to sequences of symbols does not mean that this competence is modeled by *computable* mappings (Lasersohn 2009). We had better acknowledge that one single formal system couldn't operate on all levels of explanation of our linguistic minds. This is an instance of the paradox that Gödel discovered: systems of knowledge are formal and consistent only if they are not holistic and totalitarian. If artificial intelligence is bound to stay within the limits of mathematics, grammar certainly is too. Grammar, even computational grammar, can compute a lot, but not everything that is true of grammatical objects. The study of language is not witchcraft but normal business, after all.