Formalizing the minimalist program

Veenstra, Mettina Jolanda Arnoldina

Chapter 10

Concluding remarks

The objective of this thesis is to provide a formalization of a minimalist description of a small fragment of Dutch. The fragment that is described is outlined in Section 3.4. Although the Minimalist Program is still in development precise definitions of the theory in its current stage of the development will be of use for linguists working inside and outside the Minimalist Program.

As a first attempt to formalization two small implementations in Prolog were made. These implementations are described in Chapter 2.

The first implementation, which is outlined in Section 2.1, gives a survey of the two structure-building operations Merge and Move. This implementation reveals that the two structure-building operations the Minimalist Program presupposes actually both consist of more disjunctively specified sub-cases. The structure-building operation Merge has three sub-cases: tree insertion in the complement position, tree insertion in the specifier position, and lexical insertion in the head position. The structure-building operation Move has two sub-cases: head movement and to-specifier movement. Furthermore a new definition of the Move operation is given. This definition states that the moved element in a Move operation has to be contained in the complement domain of the head of the target tree, not in the target tree as the original definition says. From this new definition we can derive that movement to the complement position is impossible. An element (tree) cannot be moved to a position that contains the tree from which the moved element must originate.

The second implementation, which is outlined in Section 2.2, is a head-corner parser for a fragment of the Minimalist Program. I argue that because of the nature of the structure-building operations of the Minimalist Program head-corner parsing is a suitable parsing technique

for the Minimalist Program. The idea of head-corners from head-corner parsing resembles closely the idea of target trees from the Minimalist Program, as we see in Subsection 2.2.2.

In the Chapters 4 through 9 the actual formalization of the minimalist description of the fragment of Dutch is given. The formalization is written in the formal-specification language AFSL. Afterwards, the entire formalization was validated by implementing it in Prolog. The formalization is declaratively stated, not derivationally as the Minimalist Program itself: it describes which trees are correct according to the Minimalist Program.

In Chapter 4 I give a description of trees. By considering minimalist trees as some kind of directed graphs, we can omit indices. The function `ConnectionTarget` represents connections between nodes. It is used instead of indices to indicate movements within trees.

In Chapter 5 minimalist ideas on features are formalized. I argue that the notion 'feature structure' be introduced in order to be able to treat the features of a node as a unit, although feature structures are not applied in the Minimalist Program. Furthermore I introduce some new features, among which the features [object], [subject], [compcat] and [speccat]. The first two features are used to refer to the object and the subject features of the verb. Since a verb must agree with both its subject and its object (if it is a transitive verb), we need a way to separate the subject agreement features from the object agreement features of the verb. Furthermore, the verb assigns case to the object. This case feature is also, as the agreement feature, a part of the value of the [object] feature. The features [compcat] and [speccat] are introduced to be able to implement subcategorization. The feature [compcat] indicates the category of the complement of a given head, while the feature [speccat] indicates the category of the specifier of a given head. Subcategorization receives no explicit mention in the Minimalist Program, but it turns out to be vital for the formalization. It is important that there is a way to represent what kind of complement or specifier a given head may select, because otherwise the formalization would for instance allow transitive verbs to behave like intransitive verbs, by not forcing transitive verbs to select both a subject and an object. This is especially essential in Zwart's version (and Chomsky's 1995 version) since in this version the derivation is not guided by the features of the lexical heads in a sentence. In Chomsky's 1993 version all lexical heads need to check all their formal features. Therefore the object features of the verb will require the verb to select an object to check against. Furthermore, subcategorization proved to be essential for the word order of sentences. In Chapter 5 I also give an exact definition of the notion 'feature checking'. I argue that the features of

a lexical constituent can only be checked against the features of a functional constituent if the feature structure belonging to the lexical constituent contains at least as many feature-value pairs as the feature structure belonging to the functional head.

In Chapter 6 I describe the way the lexicon is treated in the Minimalist Program. In Zwart's version of the Minimalist Program there are two lexicons. The first lexicon, which I call the prelexicon, is consulted when a lexical item enters the derivation. The second lexicon, which I call the postlexicon, is consulted after the derivation (at PF) to obtain the phonological features of a lexical item. I argue that the lexicon which is consulted at the beginning of the derivation may not contain underspecified feature structures because of the nature of the feature checking operation. Checking is only possible if a certain feature is present in a given functional head as well as in the lexical constituent that checks its features against it. Hence, we cannot indicate that any possible value for a given feature name can be chosen by not representing it at all (under-specification).

$\overline{\text{X}}$-Theory is described in Chapter 7. In the $\overline{\text{X}}$-rules that are specified in the formalization only bar-levels play a role, while in the original $\overline{\text{X}}$-rules of the Minimalist Program category features also play a role (for instance: $\overline{\text{X}}$ $\rightarrow$ X, YP (where X and Y represent categories)). In the formalization, feature percolation, including the percolation of the category feature, is taken care of separately. Feature percolation is based on the $\overline{\text{X}}$-rules, since features percolate up from the head with bar-level zero to higher bar-levels. The positions of specifiers, complements, heads and adjuncts are explicitly defined in the $\overline{\text{X}}$-module. This module seems to be the right location to represent this type of knowledge since in the literature $\overline{\text{X}}$-rules often implicitly define notions such as 'specifier' and 'complement'. Furthermore I argue that the two-level $\overline{\text{X}}$-Theory as applied in Zwart's version and Chomsky's 1995 version of the Minimalist Programs is problematic. $\overline{\text{X}}$-Theory and the theory of movement (or chains in our case) are mutually dependent. In the new version of $\overline{\text{X}}$-Theory heads do not always need to project, but without projection we cannot maintain the notions 'complement domain' and 'checking domain'.

In Chapter 8 I show that it is not problematic to treat head movement in a representational way by considering traces to be copies (contra Rizzi [Riz90] and Brody [Bro95]). Furthermore I argue that a [determiner] feature is needed to distinguish the treatment of pronouns and interrogative words on the one hand from nouns on the other hand with respect to the selection of determiners. The former group requires an 'empty' D (i.e. a D with no phonological content), while nouns, except for the plural forms, require a D with lexical content.

In Chapter 9 the interfaces (LF and PF) of the Minimalist Program are

described. The main result of the formalization of the interfaces is the discovery that an additional lexicon, which contains templates for all types of sentences covered by the formalization, is needed for Zwart's version of the Minimalist Program. Since LF in Zwart's framework is reached when all functional heads in a tree checked all their features, I needed something to make sure that an LF-tree always contains the required functional projections. Otherwise an LF-tree consisting of only a VP could be approved of by the formalization. In Chomsky's 1993 framework, lexical heads needed to check all their features before LF, and to check their features they require functional heads.

Suggestions for future research The formalization described in this work shows that a more formal approach to minimalist ideas leads to clearer definitions and sometimes to the discovery of inconsistencies and incompleteness. Therefore I think it can be interesting and useful to develop tools with which linguists, for instance, can compare several solutions to the same problem or can test the result of a change in a definition. The formalization described here could be used as a basis for a parser that could serve as such a tool.

Furthermore, research on the comparison of aspects of the Minimalist Program with other linguistic theories might be of interest. The formalization presented here could help with such a comparison. For example, the formalization lead to the conclusion that under-specification in the lexicon is not allowed in the Minimalist Program because of the nature of feature checking, although under-specification is common in feature-based theories such as HPSG.