Formalizing the minimalist program
Veenstra, Mettina Jolanda Arnoldina

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
1998

Citation for published version (APA):
Chapter 7

X-Theory

7.1 Introduction

X-Theory is a part of Chomskyan linguistics that is intended to simplify phrase structure rules. Phrase structure rules are rewrite rules that indicate how trees that represent phrase structure can be built.

In Section 1.2 we saw that X-Theory was based on proposals by Chomsky [Cho70]. He removed lexical properties from phrase structure rules. The specific phrase structure rules such as in Example 7.1 are replaced by more general rules such as in Example 7.2, where X and Y are used as variables for lexical categories such as noun (N), verb (V) and preposition (P).\footnote{For applications and revisions of X-Theory: cf. [Jac??], [Stu86] and [Muy82].}

Example 7.1

\begin{align*}
\text{(a)} & \quad V & \rightarrow & \quad V, \ NP \\
\text{(b)} & \quad P & \rightarrow & \quad P, \ NP \\
\text{(c)} & \quad N & \rightarrow & \quad N, \ PP
\end{align*}

Example 7.2

\[
X \rightarrow X, YP
\]

Chomsky [Cho93] considers X-Theory to be an independent set of principles, which the operations Merge and Move should consult. These two operations, also known as Generalized Transformations, create phrase structure. Both Merge and Move combine two trees (\(\alpha\) and \(\beta\)) into one (\(\gamma\)) by projecting one of the two trees (say \(\alpha\)). \(\gamma\) is the mother of \(\alpha\) and \(\beta\) and
\( \gamma \) has the same category as \( \alpha \). The newly constructed tree \( \gamma \) must obey \( \text{X-Theory} \).

\( \text{X-Theory} \) as Chomsky [Cho93] applies it is given in Example 7.3. Rule (a) introduces the specifier \( (YP) \), rule (b) introduces the complement, and rule (c) introduces the head adjoined element \( (W) \), which we will refer to as adjunct in the following. Following Kayne [Kay94] I assume that the order of the constituents in the right-hand side of the rules is fixed (see also Section 1.3). Hence, the specifier always precedes the head, the head always precedes the complement and the adjunct (if present) precedes the head, as is represented in Example 7.4.

**Example 7.3**

(a) \( XP \rightarrow YP, \overline{X} \)  
(b) \( \overline{X} \rightarrow X, ZP \)  
(c) \( X \rightarrow W, X \)

**Example 7.4**

More recently, Chomsky [Cho95, Page 241ff] no longer presupposes \( \text{X-Theory} \). The properties of \( \text{X-Theory} \) are derived from minimalist assumptions. This resulted in a major revision of \( \text{X-Theory} \), and Chomsky [Cho95, Page 249] even assumes that 'one goal of the Minimalist Program seems to be within reach: phrase structure can be eliminated entirely'. But for the moment \( \text{X-Theory} \) is still a vital component of the Minimalist Program.

Following Muysken [Muy82], Chomsky [Cho95, Page 242] takes \([+maximal] \) and \([-maximal] \) to be properties that depend on relations between nodes. Within a phrase \( XP \), the head \( X \) is \([-maximal] \) and \([-projection] \). \( XP \) is \([+maximal] \) and \([+projection] \), since it is the highest level of the \( XP \) and since its features are *projected* from the head \( X \). The intermediate level, \( \overline{X} \), is \([-maximal] \) and \([+projection] \), since it is not the highest level of the \( XP \) but it is a projection of the head \( X \). Chomsky now assumes that it is possible that a head only projects once. In that case the node directly
dominating that head is [+maximal] and [+projection]. Hence, if the node
directly dominating a head is [+maximal] or [-maximal] depends on whether
its mother is a projection of the same head or not. This idea is illustrated in
Example 7.5 and 7.6. Example 7.5 shows a tree where the node dominating
the head X is [+maximal] since this node in its turn is dominated by Y
(where X and Y stand for two distinct bundles of features). Example 7.6,
however, shows a tree where the node dominating the head X is [-maximal]
since this node in its turn is dominated by XP.

Example 7.5

```
YP
  ZP
  Y
  XP
  WP
  X
```

Example 7.6

```
YP
  ZP
  Y
  XP
  WP
  X
  UP
```

Note that the intermediate bar-level X is not present in Example 7.5
since the head X has no complement. The head X in Example 7.6 does
have a complement (UP). X has a specifier (WP) both in Example 7.5 and
in Example 7.6. Since Chomsky wants to derive X-Theory rather than
stipulating it as a set of independent principles, it is impossible to let a
head X project when it has no complement or no specifier. The operations
Merge and Move always take two trees α and β and combine them into
one tree γ by projecting either α or β.² If a head X has no complement

²See Kayne [Kay94] for the argument that trees must be binary branching.
or specifier, there is no reason for the operations Move and Merge to apply and consequently there is no possibility for X to project. A head X which does not project is at the same time [+maximal] and [-maximal].

According to Chomsky the label of a node, which is the feature bundle belonging to a node, may only contain features that represent inherent properties such as [case] and [person]. Therefore the features [maximal] and [projection] are not appropriate as a part of a label of a node. The features [maximal] and [projection] are not inherent but relational. Their values for a given node can be computed with respect to the tree (the representation) the node is part of, that is, their values are determined representationally.

The inherent (non-relational) features in the labels of nodes are the same for every projection of a given head and therefore they can be determined derivationally. This means that the label is fixed once the node it belongs to is formed by the operations Merge or Move.

Zwart [Zw97, Page 174] prefers considering phrase structure level as property that is part of the label of a node. Hence, also the phrase structure level of a node must be determined derivationally. As we saw above, Chomsky uses the features [maximal] and [projection] to express the phrase structure level of a node. The values of these features for a given node are determined representationally according to Chomsky, that is, they are determined on the basis of the position of the node in relation to the other nodes in the representation (i.e. tree) it is part of. Zwart argues that if the phrase structure level of a node is considered to be part of the label of a node, the feature [maximal] is problematic. A node γ is always [+maximal] when it is created as the root of a tree by Merge or Move. But in the next step of the derivation the features of γ could be projected again, which makes γ [-maximal]. Since the features of γ are determined derivationally, its features cannot be changed afterwards. Zwart assumes that the feature [maximal] is superfluous. By just applying the feature [projection] to indicate the phrase structure level of a node, we obtain an X-structure with only two levels instead of three, which has been proposed before (see for instance [Stu85, Hel91, Hoe91, Zw92, Kay94]). In this structure phrases are [+projection] and heads are [-projection].

Zwart [Zw97, Page 175] summarizes the possible phrase structures in the following law (where α and β are categorial features and m, n = [+/-projection]):

Example 7.7

---

3Chomsky [Cho95, Page 249] does not expect this assumption to cause any problems. As an illustration of items that must be both [+maximal] and [-maximal] Chomsky [Cho95, Page 249] mentions clitics.
\[ \alpha^n + \beta^n = \alpha^n \]

The law presented in Example 7.7 yields the phrases in Example 7.8, not taking into account the relative order of the daughters.

**Example 7.8**

\[ \text{(a) } XP \quad \text{(b) } X \quad \text{(c) } XP \quad \text{(d) } X \]

In practice the law in Example 7.7 predicts one phrase, (d), that does not seem to occur. Zwart claims that (d) is ruled out independently. It is supposed that (d) is either a complementation or an adjunction structure. In the first case, Y is the head and XP is the complement. Y, being a head, should project its categorial features. This is not the case and therefore (d) is excluded as a complement structure. In the second case, XP is supposed to be the head, Y should check its features against the head X of XP. For a checking relation we need Y to be adjoined to X like in (b). This is not the case in (d) and consequently (d) is excluded as an adjunction structure.

In the next section, I will present \( \overline{X} \)-Theory as it is formalized in the formalization. Note that neither Chomsky’s [Cho93] nor Zwart’s [Zwa97] version of \( \overline{X} \)-Theory as presented in this section is formalized. The reasons for preferring the traditional \( \overline{X} \)-Theory (where heads always project from X to \( \overline{X} \) to XP, independent of the presence of complements or specifiers) over the newer versions presented in this section are described in Section 7.3.

### 7.2 The formalization

In this section I will outline the formalization of \( \overline{X} \)-Theory. Mainly, we are dealing here with \( \overline{X} \)-Theory as described by Chomsky [Cho93]. Only at some points I decided to apply ideas from Chomsky’s 1995 framework or Zwart’s 1997 framework if this made the formalization more elegant.

The \( \overline{X} \)-module describes, among other things, which binary branching trees are allowed with respect to the bar-levels (i.e., phrase structure levels) of the nodes of trees. In this module, restrictions on trees rather than phrase structure rules are defined. A tree \( t \) satisfies \( \overline{X} \)-Theory if the boolean function \( \text{Xbar } t \) (see Definition 7.1) yields \text{True}. The functions \( \text{XBarLevels} \), \( \text{XFeatures} \) and \( \text{XRestrictions} \) which are consulted in the function \( \text{XBar} \) will be discussed in detail below. Here it will be sufficient to explain why these three functions are consulted in the function \( \text{XBar} \).
In the function XBarLevels the traditional X-rules (for instance, $X \rightarrow X$, $YP$) are formalized. Remarkable is here that the category variables $X$ and $Y$ are not mentioned in XBarLevels. This part of the X-rules is dealt with in the function XFeatures.

The function XFeatures formalizes feature percolation within projections of heads. Heads come from the lexicon as a feature structure. All non-heads receive their features from the heads they belong to. The traditional X-rules only (implicitly) deal with the percolation of category features by always mentioning one category value (e.g. $X$) twice in an X-rule: once at the left-hand side and once at the right-hand side of the arrow. Features other than category features were left out of consideration. In the formalization it turned out that it is convenient that all the features from the head (not only the category feature) are available at all levels. In this way it is, for instance, possible to refer to the features of an XP or to the features of the mother of any particular node in a tree, and not only to the features of lexical heads.

The significance of separating the features that describe the phrase structure level of a node (here represented by the feature BarLevel, and in Chomsky's 1995 framework represented by the features [maximal] and [projection]) from all the other features was discovered independently of Chomsky [Cho95]. As we saw in Section 7.1, Chomsky considers phrase structure level to be relational, while he considers all other features such as [number], [gender] and [tense] to be inherent. In the formalization BarLevel turned out to be an exceptional feature since it was the only feature that projections did not seem to inherit from their head. Furthermore, the feature BarLevel proved to be essential to describe the way features percolate from heads to their projections.

The function XRestrictions specifies a residue. It describes a kind of feature checking which is not invoked by movement. In the Minimalist Program feature checking and movement are intimately linked. Movement of constituents is only allowed if it is required for purposes of feature checking. During the formalization process there turned out to be a type of feature checking which is not only linked with the operation Move, but in some cases also with the operation Merge. This type of feature checking is described in the function XRestrictions. The kind of feature checking I am referring to here is required in order to let a head, lexical or functional, select the right arguments. These arguments can either manifest themselves as specifiers or complements, as we will see below.

After the more extensive description of the functions XBarLevels, XFeatures and XRestrictions I will discuss the functions in which specifiers, heads etc. are given a position within projections. This is not an official part of X-rules, but I will argue that the X-module is the suitable
module for this type of information.

**Definition 7.1**

\[
\text{FUNC } \text{XBar} : \text{TreeS} \rightarrow \text{BoolS}
\]

\[
\text{AXIOM } \text{XBar tr} \leftarrow \text{XBarLevels tr} \land \text{XFeatures tr} \land \text{XRestrictions tr}
\]

**Bar-levels** In the function \text{XBarLevels} the bar-levels of nodes are given on the basis of the bar-level of their mother. The function is based on the following definition I formulated (see Definition 7.2 for an excerpt from the formalization and Example 7.9 for a picture of a tree):

- If the bar-level of a node nd is not undefined, then the bar-level of nd is either 0, 1 or 2 (first axiom in Definition 7.2). The bar-level of an empty leaf is undefined.

- If the bar-level of a node nd is 2, then the bar-level of its left daughter is 2 or its left daughter is empty. Furthermore, the bar-level of the right daughter of nd is 1 (second axiom in Definition 7.2). Note that the implication can also be read in the other direction.

- If the bar-level of a node nd is 1, then the bar-level of its left daughter is 0. Furthermore, the bar-level of the right daughter of nd is 2 or the right daughter nd is empty (third axiom in Definition 7.2). Note that the implication can also be read in the other direction.

- If the bar-level of a node nd is 0, then nd is a nonempty leaf, or the left daughter of nd has bar-level 0 and the right daughter of nd has bar-level 0 and is a nonempty leaf (fourth axiom in Definition 7.2). Note that the implication can also be read in the other direction.

**Definition 7.2**

\[
\text{FUNC } \text{BarLevel} : \text{NodeS} \rightarrow \text{PARTIALNatS}
\]

\[
\text{AXIOM } \text{BarLevel nd }\neq \text{ Undef} \Rightarrow (\text{BarLevel nd}) \in \text{Set}(0,1,2)
\]

\[
\text{AXIOM } \text{BarLevel nd }= 2 \leftarrow \text{EmptyLeaf LeftDaughter nd} \lor \text{BarLevel LeftDaughter nd }= 2
\]
AXIOM BarLevel \( nd = 1 \) \iff BarLevel LeftDaughter = 0
And ( EmptyLeaf RightDaughter \( nd
\) Or BarLevel RightDaughter \( nd = 2 \) )

AXIOM BarLevel \( nd = 0 \) \iff NonEmptyLeaf \( nd
\) Or ( BarLevel LeftDaughter = 0
And NonEmptyLeaf RightDaughter \( nd \) )

FUNC XBarLevels : TreeS \to BoolS

AXIOM XBarLevels \( tr \) \iff BarLevel \( tr \) \neq Undef

Example 7.9

Note that the illustration in Example 7.9 is just a possible tree. More
or fewer adjunctions than the depicted amount of two are permitted. And
of course it is possible to have a head with no adjunctions at all.

Because a head \( X \) requires two projections (\( X \) and \( XP \)), the predicate
EmptyLeaf is required. For instance, even if a phrase does not contain a
complement, we need an \( X \)-projection, and this projection can only arise
by applying one of the structure-building operations. Since the operations
Merge and Move combine two trees into one, we need a substitute for the
non-existing complement. The predicate EmptyLeaf is used as a substitute.

The feature Category does not occur in the function BarLevel. This
is not necessary because the three rules (axioms) of the function BarLevel
are mutually exclusive without referring to the feature Category: there are
no nodes \( nd \) that could be described by two different rules.
The need to refer to left and right daughters emerges since I accept Kayne's [Kay94] conclusion that there is only one possible way to arrange specifiers, heads, complements and adjuncts (see Section 1.3). For instance, the specifier must precede the head. Therefore, a node with BarLevel 2 must have a node with BarLevel 2 or an EmptyLeaf (the specifier) as its left daughter, and a node with BarLevel 1 (which dominates the head) as its right daughter.

The definition of BarLevel is recursive. If we want to compute the bar-level of the root of a tree, we have to compute the bar-levels of all the nodes of the tree. It is possible to compute the bar-level of a node of a tree because trees are finite, as is specified in the tree module. In the fourth axiom of the function BarLevel we see that lexical heads always adjoin to the left. Namely, the bar-level of the left daughter is 0, which according to the same axiom means that it can either be a nonempty leaf or a node with a left and a right daughter. The latter possibility is required because adjuncts themselves can have an adjunct as a daughter. Therefore the right daughter in the fourth axiom can never be an adjunct: it must be a nonempty leaf.

The first axiom in Definition 7.2 implies that all nodes nd that do not have a bar-level 0, 1 or 2 do have a bar-level that is undefined. For instance, if it is impossible to compute the bar-level of a certain node or if a node is an empty leaf, then its bar-level is undefined. It is impossible to compute the bar-level of a node nd which is inconsistent with respect to the \( X\)-rules defined in the first four axioms in Definition 7.2. Therefore the bar-levels of nd and all the nodes that dominate nd are undefined.

The function XBarLevels at the bottom of Definition 7.2 accounts for the bar-level of the root of a tree tr. This boolean function fails if the bar-level of the root of tr is undefined. The bar-level of the root of tr is undefined (as we saw in the above paragraph) when the root of tr is an empty leaf or if the root of tr is inconsistent with respect to the \( X\)-rules or if it dominates a node which is inconsistent with respect to the \( X\)-rules.

Features The \( X\)-module also describes the distribution of features in trees. The function XBar tr in Definition 7.1 calls the function XFeatures (see Definition 7.3) to ensure that the features in tr are correctly percolated. Features have such a prominent role in the Minimalist Program because of the importance of feature checking for deriving word order variation, and so they need to be connected with \( X\)-Theory. Therefore features are percolated

---

4An illustration of a lexical head \( V \) which adjoins to another head \( \text{AgrO} \) is given on Page 20 in Example 1.15. \( V \) is called the adjunct in this example.

5An illustration of an adjunct which has an adjunct as its daughter can be found in Example 1.17 on Page 22, where the adjunct \( \text{AgrO} \) has the adjunct \( V \) as its daughter.
up from the head in the formalization described here, so that features are available at all levels of the projections. This is desirable because we can then retrieve the features of both trees with \texttt{BarLevel 0} and \texttt{BarLevel 2} that move for feature checking in their root.

The X-module is the appropriate module to deal with the distribution of features because features are connected with the units that X-Theory describes (XPs). In every XP the head is the node from which the projections inherit their features. Within a tree \( \text{tr} \) with \texttt{BarLevel 2} features percolate up from the head of \( \text{tr} \) to its mother (with \texttt{BarLevel 1}) and its grandmother (with \texttt{BarLevel 2}).

In ‘traditional’ X-Theory Category was the only feature that was percolated up from the head to the head’s mother and grandmother. This was done by always mentioning the same category variable (for example: X) twice in every X-rule, once on the left-hand side and once on the right-hand side (for example: \( X \rightarrow X \text{ Complement} \)). In the version of X-Theory formalized here, on the other hand, the percolation of more features than only category features is proposed, because this is a requirement for the success of the rest of the formalization. Namely, features need to be available at all levels of a projection.

Definition 7.3 shows the formalization of feature percolation. The nodes \( n_d \) of a tree \( \text{tr} \) are divided into two groups: leaves and non-leaves. The features of a non-leaf \( n_d \) are equivalent to the features of its head.\(^7\) The feature structure belonging to a leaf \( n_d \) (if it is nonempty) must be a member of the lexicon.

The requirement that the features of a nonempty leaf must be a lexical item is our equivalent of lexical insertion. The formalization can judge whether a tree is built according to the Minimalist Program. It cannot build trees with the operations Merge and Move in combination with lexical insertion. The formalization deals with abstract trees that might have been constructed by Merge and Move. The only way to check whether lexical insertion took place correctly is by determining if all heads from the tree are taken from the lexicon.

The fact that \texttt{BarLevel} is not considered to be an inherent feature is very advantageous for the readability of the function \texttt{XFeatures}. If the feature \texttt{BarLevel} had been a part of the feature structure belonging to a node, then the features of the head would not have been equal to the features of its projections. Namely, the \texttt{BarLevel} of the head is 0 while the \texttt{BarLevel} of a

\(^6\)In Generalized Phrase Structure Grammar this is called the Head Feature Convenion [Gas82] and in Head-driven Phrase Structure Grammar this is called the Head Feature Principle [see [PS04]]. These theories share with the current formalization the percolation of features connected with lexical heads.

\(^7\)The definition of the notion ‘head’ is given further on in this section, together with the definitions of complements, specifiers and adjuncts.
projection is either 1 or 2, dependent on the projection being intermediate (X) or maximal (XP). Therefore, the feature BarLevel would have been separated from the rest of the features if we treated it as an inherent feature, and this would imply an operation of removing the BarLevel feature from the rest of the features within the function XFeatures.

Something similar holds for the feature Word if we treat it according to Chomsky's 1993 framework. In this framework only leaves carry the feature Word, because leaves are the only nodes of trees where words are depicted. In Chomsky's 1995 framework we see that words are present in all the projections of a head. The ideas with respect to the feature Word from the 1995 framework are preferred over those in the 1993 framework since the former ideas simplify the function XFeatures. If the Word feature would have been present in the head of a phrase, it would have to be separated from the rest of the features when the features of the head are percolated up to the projections. In the way the feature Word is treated in this formalization, this extra operation is not necessary.

**Definition 7.3**

\[
\text{FUNC } \text{XFeatures} : \text{TreeS} \rightarrow \text{BoolS}
\]

\[
\text{AXIOM } \text{XFeatures tr} \\
\iff \forall \text{nd} \\
\quad \text{nd NodeOf tr} \\
\quad \Rightarrow (\text{Not IsLeaf nd} \\
\quad \Rightarrow (\text{Features nd} = \text{Features Head nd} \\
\quad ) \\
\quad ) \\
\quad (\text{NonEmptyLeaf nd} \\
\quad \Rightarrow (\text{Features nd}) \text{ In Lexicon} \\
\quad ) \\
\quad )
\]

In a construction such as the tree in Example 7.10 all the right daughters, except for TP, are heads (see further on in this section) and therefore the mothers inherit their features. For instance, the higher AgrO inherits the features of the lower AgrO since the latter is a head.

**Example 7.10**

```
AgrS
  /   
AgrS  TP
  /  
AgrS  
  / 
AgrO  
  /  
  V   AgrO
```

---

7.2. *THE FORMALIZATION*
Additional restrictions In the minimalist framework the term ‘feature checking’ is associated with movement. In the boolean function XRestrictions a kind of feature checking is formalized which cannot be referred to as ‘feature checking’, because it is not always associated with movement (see Definition 7.4).

Definition 7.4

\[
\text{F\textsc{unc}} \quad \text{XRestrictions} : \text{Tree} \rightarrow \text{Bool}
\]

\[
\text{A\text{x\textit{im}}} \quad \text{XRestrictions } \text{tr} \\
\quad \quad \quad \quad \Rightarrow \text{FA}\text{ULL } \text{nd} \\
\quad \quad \quad \quad \quad ( \text{nd NodeOf tr} \\
\quad \quad \quad \quad \quad \quad \text{And BarLevel } \text{nd} = \text{Two} \\
\quad \quad \quad \quad \quad \quad \quad \Rightarrow ( \text{Value}[\text{Category}] \text{Complement } \text{nd} \\
\quad \quad \quad \quad \quad \quad \quad \quad = \text{Value}[\text{CompCat}] \text{nd} \\
\quad \quad \quad \quad \quad \quad \quad \quad \quad \text{And} \quad \text{Value}[\text{Category}] \text{Specifier } \text{nd} \\
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad = \text{Value}[\text{SpecCat}] \text{nd} \\
\quad \quad \quad \quad \quad \quad \quad )
\]

The axiom in Definition 7.4 indicates that all nodes \text{nd} in a tree \text{tr} that have BarLevel 2 must have:

- a complement that has the same category as the value of the feature CompCat (if the complement is undefined, the value of the feature CompCat must also be undefined)

- a specifier that has the same category as the value of the feature SpecCat (if the specifier is undefined, the value of the feature SpecCat must also be undefined)

Only nodes with BarLevel 2 are taken into account because these are the only nodes that can have a complement as well as a specifier. As we will see further on in this section, nodes with BarLevel 1 do not have a specifier because the position of the specifier is not dominated by the node with BarLevel 1 (see Example 7.13). A node with BarLevel 0 neither has a specifier nor a complement because the complement and the specifier positions are not dominated by the node with BarLevel 0 (see Example 7.13).

The features CompCat and SpecCat respectively indicate the category of the complement and the specifier a given head selects. The features CompCat and SpecCat, as all other features of the head, are percolated up to the projections of the head. Therefore it is possible to check the category features of the complement and the specifier of \text{nd} (with BarLevel 2) against its CompCat and SpecCat values, respectively.
The phenomenon described in the above paragraph could be referred to as subcategorization. In Chomskyan linguistic theory, subcategorization is linked with the assignment of thematic roles (cf. [Cho95], [HK93]). For instance, whether a verb is transitive or intransitive is not a primitive property: it follows from the meaning of the verb. The verb see expresses an activity that involves two participants: one active participant who sees, and an inactive object or participant that is looked at. The thematic roles ($\theta$-roles) that are assigned by the verb see are Agent (for the active participant) and Patient (for the inactive object or participant).

In the Minimalist Program [Cho95, Page 312ff], the role of $\theta$-Theory is to make sure that derivations where not all arguments are assigned $\theta$-roles will not converge (succeed). $\theta$-roles are not associated with movement but with merger because they are assigned to a lexical item when it is inserted in a structure, and consequently lexical insertion is always merger and never movement. Because $\theta$-role assignment is connected with lexical insertion, $\theta$-roles are always assigned in the lowest position of a chain. For instance, a subject always receives its $\theta$-role when it is in the specifier position of the VP, not when it is in the specifier position of T or AgrS, or in an even higher position. If the subject entered a derivation in the specifier position of, for instance, T, this derivation would be more economical than a derivation where the subject enters the derivation in the specifier position of VP, because the latter derivation involves more steps. However, the former derivation would not converge because the object never occurred in a configuration where it was assigned a $\theta$-role.

Subcategorization frameworks are not described within the Minimalist Program. However, subcategorization turns out to be vital for the formalization. We need a way to represent what kind of complement or specifier a certain head may select, otherwise the formalization would for instance allow transitive verbs to behave like intransitive verbs, by not forcing intransitive verbs to select both a subject and an object. This is especially essential in Zwart’s 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 it proved to be essential for the word order of sentences that functional projections always appear in the same order. Hence, also functional heads must contain the features CompCat and SpecCat. For instance, if the formalization allowed the functional head AgrS to select a CP complement instead of allowing the functional head C to select an AgrSP complement, the word order of a subordinate clause like the one in Example 7.11 would end up with the complementizer (that) in the wrong position as in Example 7.12.
Example 7.11
that she kisses him

Example 7.12
she kisses that him

Therefore the function XRestrictions is applied to both lexical and functional projections. Most functional projections (AgrOP, CP etc., but not DP) take DP specifiers which arrive there by movement. Furthermore, functional heads are always 'united' with their complements by the operation Merge, since, as we saw in Chapter 2, movement to complement positions is impossible. In the lexical domain, that is within the VP, both the specifier and the complement are created by the operation Merge, since in the lexical domain lexical insertion takes place. Of course lexical insertion is Merger by definition.

Summarizing, we can say that the function XRestrictions specifies a kind of feature checking which is not exclusively associated with movement, but which we need in order to approve only of sentences with a correct word order, and a correct meaning (in the sense that, for instance, in transitive verbs never select a complement).

Stabler [Sta96, Page 108] proposes a comparable approach to the selection of complements and specifiers. Both functional and lexical heads are associated with a list of features which may contain either zero, one or two selection features. The selectional features indicate the category of the specifier and/or complement the head selects. If a feature list contains two selectional features, the first represents the complement and the second the specifier.

Specifiers, heads, complements and adjuncts The X-module also identifies specifiers, heads, complements and adjuncts within XPs (if they exist). This module is a natural location to represent this information because X-rules in the literature often contain these notions (for instance: XP ֒→ Spec(Spec) X). Sometimes the information is not given in the rules (for instance: XP ֒→ YP X). In such cases the positions of specifiers, heads, complements and adjuncts are discussed elsewhere, or this is supposed to be familiar to the reader.

Definition 7.5 consists of four different axioms: one for each bar-level (2, 1 and 0) plus one for cases in which the bar-level is undefined. The functions Specifier, Complement, Head and Adjunct are all partial functions. This implies that the functions are undefined for certain trees. For instance:

---

8 cf. [Hae91, Page 96]
For a tree $tr$ with $\text{BarLevel } 2$, the specifier can only be undefined if the specifier position (see the tree in Example 7.13) is 'occupied' by an empty leaf (see the second conjunct in the first axiom in Definition 7.5). If the specifier position is not occupied by an empty leaf, $tr$ has a specifier because the root of $tr$ dominates the specifier position (see the first conjunct in the first axiom in Definition 7.5).

A tree with $\text{BarLevel } 1$ does not contain a specifier because the node with $\text{BarLevel } 1$ (i.e. $\overline{X}$) does not dominate the specifier position (see Example 7.13). This phenomenon is accounted for by the first conjunct of the second axiom in Definition 7.5.

Both the head and the adjunct in the tree in Example 7.13 do have undefined heads and adjuncts because neither of the two dominates a head or adjunct. In Definition 7.5 this phenomenon is accounted for by the third conjunct of the third axiom: for leaves with $\text{BarLevel } 0$ adjuncts and heads are undefined.

A tree with an undefined bar-level, i.e. an empty leaf, does not have a specifier nor a complement nor a head nor an adjunct for the obvious reason that it does not dominate a specifier position or a complement position etc. (see the fourth axiom in Definition 7.5).

**Definition 7.5**

\[
\begin{align*}
\text{FUCB } & \text{Head : TreeS } \rightarrow \text{PARTIAL TreeS} \\
\text{FUCB } & \text{Complement : TreeS } \rightarrow \text{PARTIAL TreeS} \\
\text{FUCB } & \text{Adjunct : TreeS } \rightarrow \text{PARTIAL TreeS} \\
\text{FUCB } & \text{Specifier : TreeS } \rightarrow \text{PARTIAL TreeS}
\end{align*}
\]

**AXIOM**

\[
\begin{align*}
\text{BarLevel } tr & = 2 \\
\implies & ( \text{Not EmptyLeaf LeftDaughter } tr \\
\implies & \text{Specifier } tr = \text{LeftDaughter } tr ) \\
\text{And } & ( \text{EmptyLeaf LeftDaughter } tr \\
\implies & \text{Specifier } tr = \text{Undef} ) \\
\text{And } & \text{Complement } tr = \text{Complement RightDaughter } tr \\
\text{And } & \text{Head } tr = \text{Head RightDaughter } tr \\
\text{And } & \text{Adjunct } tr = \text{Adjunct RightDaughter } tr
\end{align*}
\]

**AXIOM**

\[
\begin{align*}
\text{BarLevel } tr & = 1 \\
\implies & \text{Specifier } tr = \text{Undef} \\
\text{And } & ( \text{Not EmptyLeaf RightDaughter } tr \\
\implies & \text{Complement } tr = \text{RightDaughter} ) \\
\text{And } & ( \text{EmptyLeaf RightDaughter } tr
\end{align*}
\]
The definitions of specifiers, complements, heads and adjuncts are recursive. For instance, if the root of a certain tree (say tr with BarLevel 2) does not have a daughter that is a complement position, but if the root does dominate a complement position, then the complement is defined as...
the complement of one of the daughters (see the third conjunct of the first axiom in Definition 7.5). If the right daughter of $tr$ is an empty leaf, the bar-level of the right daughter is undefined and therefore the complement of $tr$ is undefined (see fourth axiom).

Sometimes the definitions seem to allow incorrect trees. For instance:

- In the first axiom in Definition 7.5 the left daughter of $tr$ can either be an empty leaf (second conjunct) or not an empty leaf (first conjunct). If the left daughter is not an empty leaf this can either imply that it is a nonempty leaf or a non-leaf (i.e. a tree with more than just a root). If the left daughter is a nonempty leaf, $tr$ would not be a correct tree in a theory where it is assumed that all heads should project. We also assume that only XPs can serve as specifiers and complements, and XPs can never be (nonempty) leaves. Still we leave the definition as it is because this is not the place to prohibit specifiers that are leaves. Specifiers that are leaves are excluded by the function $BarLevel$. In this function (second axiom in Definition 7.2) we claim that specifiers (left daughters of trees with $BarLevel$ 2) must be trees with $BarLevel$ 2. And from the same axiom we can deduce that trees with $BarLevel$ 2 can never be leaves as their roots must always have a left daughter (possibly empty) and a right daughter (with $BarLevel$ 1).

- The left daughter of a tree $tr$ with $BarLevel$ 1 can either be an empty leaf, a nonempty leaf or a non-leaf (i.e. a tree with more than just a root). However, only two of the three possibilities are correct according to the Minimalist Program: nonempty leaves and non-leaves. An empty left daughter of a tree with $BarLevel$ 1 is incorrect as is specified in the function $BarLevel$ (see Definition 7.2), and therefore we do not have to worry here about the incorrectness of empty nodes as a left daughter of trees with $BarLevel$ 1 (see last two conjuncts of the first axiom of Definition 7.5).

A projection can be either deeper or shallower than the tree in Example 7.13. The difference occurs at $BarLevel$ 0.

The projection is deeper in the case of branching adjuncts. An adjunct, which always is the left daughter of a tree with $BarLevel$ 0 and which has a sister that is a head, can itself also have an adjunct as a left daughter (and a head as its right daughter) etc. This structure is needed for head movement. For instance, if the head of the VP moves to AgrOP to check its features, it adjoins to AgrO. If the head moves further to TP the whole adjunction construction from AgrOP, that is, the whole phrase that is dominated by the highest node with $BarLevel$ 0, adjoins to T. Again, if the head moves further, the whole left daughter of T adjoins to AgrS. In Example 7.10 the depth of the adjunct in AgrSP is shown. Heads never branch. In the fourth
axiom in Definition 7.2 we see that right daughters of nonempty heads with \texttt{BarLevel} 0 always are empty leaves.

The projection is shallower than the tree in Example 7.13 when the left daughter of a tree with \texttt{BarLevel} 1 is a nonempty leaf. The third axiom in Definition 7.5 shows that trees with \texttt{BarLevel} 0 can be nonempty leaves. The second axiom in the same example shows that if the left daughter of a tree with \texttt{BarLevel} 1 is a nonempty leaf, then this nonempty leaf must be a head.

Except for being able to refer to the specifier, complement, head or adjunct of a certain tree, it is sometimes necessary to refer to a certain tree as a specifier, complement, head or adjunct. Therefore the functions in Definition 7.6 are defined. The functions are rather straightforward: a tree \texttt{tr1} is a specifier if there is a tree \texttt{tr2} of which \texttt{tr1} is the specifier etc.

We need the function \texttt{IsAdjunction} to be able to refer to the mothers of adjuncts. This is convenient with respect to head movement where the moving element grows with each step (see Example 7.10), because in the second move it takes along the construction it adjoined to in the first move, etc. Because an adjunction is the mother of an adjunct it can never be a leaf (see fifth axiom in Definition 7.6).

\textbf{Definition 7.6}

\begin{verbatim}
FUNC IsSpecifier : TreeS -> BoolS
FUNC IsComplement : TreeS -> BoolS
FUNC IsHead : TreeS -> BoolS
FUNC IsAdjunct : TreeS -> BoolS
FUNC IsAdjunction : TreeS -> BoolS

AXIOM IsSpecifier tr1
   <=> EXISTS tr2 (Specifier tr2 = tr1)

AXIOM IsComplement tr1
   <=> EXISTS tr2 (Complement tr2 = tr1)

AXIOM IsHead tr1
   <=> EXISTS tr2 (Head tr2 = tr1)

AXIOM IsAdjunct tr1
   <=> EXISTS tr2 (Adjunct tr2 = tr1)

AXIOM IsAdjunction tr
   <=> BarLevel tr = 0
   And Not IsLeaf tr
\end{verbatim}
7.3 Two-level $\mathbf{X}$-Theory

In this section, I will discuss why I prefer to apply Chomsky’s 1993 version of $\mathbf{X}$-Theory instead of the later versions by Chomsky [Cho95, Page 241ff] or Zwart [Zwa97, Page 171ff], which I discussed in Section 7.1. As was mentioned in Section 7.1, $\mathbf{X}$-Theory is no longer presupposed in the later versions of the Minimalist Program. $\mathbf{X}$-Theory is now derived from minimalist assumptions. The main idea behind the new versions of $\mathbf{X}$-Theory is that $\mathbf{X}$-Theory may no longer force us to build a tree in a way that is not natural with respect to the lexical material the tree is based on. The operations Move and Merge are the only structure-building operations in the Minimalist Program. They both construct trees by combining two other trees. If we do not want $\mathbf{X}$-Theory to be independent, we do not want to assume an empty tree when a given head does not select a specifier or a complement just to be able to derive the right number of projections, as in the case of the specifier of X in Example 7.14. If $\mathbf{X}$-Theory is not independent there is no ‘right’ number of projections. For instance, if a head X selects a complement YP and no specifier, the head X only projects once, as in Example 7.15.

Example 7.14

```
< empty >
```

Example 7.15

```

In the rest of this section we will see that the idea that heads only project if they select a specifier and/or a complement is problematic. For two different reasons we will come to the conclusion that we have to assume that a head always selects at least one specifier or one complement, possibly an empty one. This makes the new version of $\mathbf{X}$-Theory so little different from the earlier three-level version, that I decided not to implement the new version of $\mathbf{X}$-Theory.

The first reason why we have to assume that a head always selects at least one specifier or complement is caused by the fact that we have to be able
to point out all the specifiers, adjuncts and complements in a tree. These positions are vital for the theory of movement, which says that movement is only possible from the complement domain of a given head to the checking domain of that same head. The complement domain consists of all the nodes that are reflexively dominated by the complement of a head X. The checking domain consists of the specifier position and the adjunct position of a head X.

The fact heads do not always have to project, namely not in case they do not select a complement nor a specifier, prevents us from making the notions ‘complement’, ‘specifier’ and ‘head’ deterministic. Since heads do not always project, complements and specifiers may appear in a tree as a leaf (see for instance the complement in Example 7.16). Of course, heads are always leaves. From the above facts we can deduce the following problematic $\Xi$-rule:

$$ [+projection] \rightarrow [-projection], [-projection] $$

The above rule is problematic, because it both describes a tree with specifier as a left daughter and a head as a right daughter (see Example 7.17), and a tree with a head as a left daughter and a complement as a right daughter (see Example 7.18). In the formalization described in Section 7.2, we could unambiguously point out specifiers and complements in a tree on the basis of the bar-levels of their mothers in combination with their being a left or a right daughter:

- a specifier is the left daughter of a node with $\text{BarLevel} = 2$
- a complement is the right daughter of a node with $\text{BarLevel} = 1$

If we apply the same idea to the $\Xi$-rules of the new version of $\Xi$-Theory, the head position gets mixed up with both the specifier position and the complement position:

- a specifier is the left daughter of a node which is $[+projection]$
- a complement is the right daughter of a node which is $[+projection]$
- a head is the left or the right daughter of a node which is $[+projection]$

---

9I use Zwart's equivalent of $\text{BarLevel}$ here, because in the formalization the $\Xi$-rules are also solely based on bar-levels.
Example 7.16

\[
\text{DP} \\
\text{D} \quad \text{N} \\
\text{the} \quad \text{woman} \\
(h\text{ead}) \quad (\text{complement})
\]

Example 7.17

\[
\text{XP} \\
\text{Y} \quad \text{X} \\
(s\text{pecifier}) \quad (h\text{ead})
\]

Example 7.18

\[
\text{XP} \\
\text{X} \quad \text{Y} \\
(h\text{ead}) \quad (\text{complement})
\]

To compensate for the fact that the head cannot be unambiguously defined, categorial features are needed to formalize $\overline{X}$-rules (see for instance Definition 7.7, second axiom, second and fourth conjunct). It is possible to distinguish heads from complements and specifiers because heads have the same category as their mother, while specifiers and complements do not. However, in this way we start a circular reasoning which is avoided when we apply $\overline{X}$-Theory as it is described in Section 7.2. The circular reasoning originates because the function $\text{IsProjection}$ (see Definition 7.7) is based on the feature $\text{Category}$ while the function $\text{XFeatures}$ (see Definition 7.3) which takes care of the percolation of features (such as $\text{Category}$) relies on bar-levels. In the traditional type of $\overline{X}$-rules (e.g. $\overline{X} \rightarrow X, Y$) there also was a variable present for the feature $\text{Category}$ but we were able to ban categorial features from the function $\text{BarLevel}$ (see Definition 7.2) in the formalization which we based on these $\overline{X}$-rules.

Another reason for categorial information in the function $\text{IsProjection}$ is the fact that without this information we cannot determine which node is the highest node of a certain projection. What formerly was an XP could now appear as an X (if it does not have a specifier or a complement). The only way of determining whether a certain node is the top level of a
projection is by looking at its mother (see first and second conjunct of the second axiom of Definition 7.7). If the mother has a different category we know that it is the top level. However, this solution is not adequate because it might result in a head of category X taking a complement of the category X. For instance, Example 7.19 shows a Larsonian structure where a verb takes a verb complement.

**Definition 7.7**

\[\text{Func} \text{ IsProjection} : \text{NodeS} \rightarrow \text{PARTIALNatS}\]

\[\text{Axiom} \quad \text{IsProjection nd} \neq \text{Undef} \quad \Rightarrow \quad (\text{IsProjection nd}) \text{ In Set}(0,1)\]

\[\text{Axiom} \quad \text{IsProjection nd1} = 1 \quad \Leftrightarrow \quad \exists \text{ nd2} \quad \begin{cases} \text{nd1} = \text{Mother nd2} \\ \text{And} \quad \text{Value[Category] nd1} \\ \neq \text{Value[Category] nd2} \\ \text{And} \quad (\text{IsProjection nd2} = 1 \\ \text{Or} \quad \text{IsProjection nd2} = 0) \\ \text{And} \quad \text{Value[Category] nd1} \\ = \text{Value[Category] Sister nd2} \\ \text{And} \quad (\text{IsProjection Sister nd2} = 1 \\ \text{Or} \quad \text{IsProjection Sister nd2} = 0) \end{cases}\]

\[\text{Axiom} \quad \text{IsProjection nd1} = 0 \quad \Leftrightarrow \quad \text{IsLeaf nd1} \quad \text{Or} \quad \exists \text{ nd2} \quad \begin{cases} \text{nd1} = \text{Mother nd2} \\ \text{And} \quad \text{Value[Category] nd1} \\ \neq \text{Value[Category] nd2} \\ \text{And} \quad \text{IsProjection nd2} = 0 \\ \text{And} \quad \text{Value[Category] nd1} \\ = \text{Value[Category] Sister nd2} \\ \text{And} \quad (\text{IsProjection Sister nd2} = 0 \\ \text{And} \quad \text{IsLeaf Sister nd2}) \end{cases}\]

\[\text{Func} \text{ XProjections} : \text{TreeS} \rightarrow \text{BoolS}\]

\[\text{Axiom} \quad \text{XProjections tr} \quad \Leftrightarrow \quad \text{IsProjection tr} \neq \text{Undef}\]

\[\text{[10] Generally, this is not supposed to be the case [see [Hoe84]].}\]

\[\text{[11] The Larsonian structure is an analysis of multi-argument verbs [Lar88].}\]
A solution to the problem sketched above would be to make the difference between heads on the one hand and specifiers and complements on the other hand unambiguous. This could be accomplished by requiring specifiers and complements to project even though their heads do not select a specifier or a complement.

Another reason why we have to assume that a head always selects at least one specifier or complement is caused by the fact that I adopt the linear ordering of specifiers, adjuncts, heads and complements that Kayne [Kay94] proposes in his Linear Correspondence Axiom (LCA). Kayne derives this linear ordering from the hierarchical structure of trees: if two subtrees A and B of a tree C, occur in C in a certain hierarchical relation (namely, A asymmetrically c-commands B), then A linearly precedes B. However, the LCA puts some requirements on X-Theory, among which the requirement that complements cannot be bare heads, since otherwise a head cannot asymmetrically c-command its complement.\(^\text{12}\)

Since one of the basic ideas behind X-Theory was to make phrase structure rules category-independent, we do not want to construct a special X-rule for categories which can serve as a complement. Hence, we must conclude, again, that all categories must project before they can be selected as a specifier or a complement.

Now we have concluded, for two different reasons, that heads must project before they can be selected by another head as a specifier or a complement, we will have a closer look at the nature of this vacuous projection. Below we will see that the assumption of a null-complement is preferred over the assumption of a null-specifier.

\(^{12}\)See [Kay94, Page 7ff] for an explanation.
Hale and Keyser [HK93] argue that also intransitive verbs take a complement. For instance, the verb *to dance* is actually an abstract V, in which the nominal head of its NP-complement (*dance*) is incorporated, hence: *to dance a dance*. This idea is applied by Stabler [Sta96, Page 108] in his derivational formalization of some minimalist ideas.

In Stabler’s formalizations, intransitive verbs are represented as projections in the lexicon (see Example 7.20 for a simplified version of the lexical item for the verb *to dance*). This is remarkable because, of course, normally the lexicon only contains heads. Stabler needs this solution since, as we saw in Section 7.2, he assumes that heads have a list of selectional features in the order object-subject. Since intransitive verbs only select a subject (i.e. specifier), Stabler needs intransitive verbs to be represented in the lexicon with the complement already included, so that the first (and only) element on the list of selectional features (D) will be interpreted as a specifier. Example 7.21 shows a simplified version of the lexical item for the transitive verb *to love*. The first feature on the list of selectional features represents the object (or complement), the second represents the subject (or specifier).

**Example 7.20**

![Diagram](..)

**Example 7.21**

![Diagram](..)

For nouns Stabler has a different solution. Nouns do not contain a null-complement in the lexicon. Stabler does not give a reason for this difference in treatment of verbs and nouns, but the reason might be that it is not necessary for the simple type of DPs he deals with in his formalization, which only contain determiners and nouns. Another reason might be that nouns never seem to select a specifier, so there is no risk of confusing the element on the list of selectional features.\(^\text{13}\)

The fact that unprojected complements do not cause problems for Stabler’s formalization lies in the fact that his approach is derivational and

\(^{13}\) See for instance [Lat97].
7.3. TWO-LEVEL \( \mathbf{x} \)-THEORY

not representational. At the moment that the head D and the unprojected N-complement merge, an arrow (\(<\)) is put in the label of their mother as in Example 7.22.\(^{14}\) The arrow indicates that the features of the label are percolated from the left daughter (D). Therefore, complements can be defined as the right daughter of a node with the label ‘\(<\)’, and heads as the left daughter of a node with the label ‘\(<\)’. Specifiers can be defined as the left daughter of a node with the label ‘\(>\)’. The mother of the specifier is a node with the label ‘\(>\)’ because the features of the mother are percolated from the right daughter (see Example 7.23).

Example 7.22

\[
\begin{array}{c}
\text{D} \\
\text{N} \\
\text{[N]} \quad \text{[]} \\
\text{the} \\
\text{car}
\end{array}
\]

Example 7.23

\[
\begin{array}{c}
\text{D} \\
\text{N} \\
\text{V} \\
\text{[N]} \quad \text{[]} \\
\text{the} \\
\text{girl} \\
\text{loves} \\
\text{D} \\
\text{N} \\
\text{[N]} \quad \text{[]} \\
\text{her} \\
\text{car}
\end{array}
\]

The fact that Stabler can apply \(\text{category}\) features to distinguish complements and specifiers from heads and projections of the head, and I cannot, lies in the fact that Stabler’s approach is derivational and my approach is representational. In a representational approach one wants to prevent incorrect structures from being approved of, while in a derivational approach incorrect \(\mathbf{x}\)-structures are prevented from being built. In Stabler’s approach, incorrect feature percolation can be prevented by requiring that Merge and Move always project one of the two trees they combine. In my

\(^{14}\)Note that Stabler cancels \(\text{i.e. deletes}\) the selectional feature \(N\) of the determiner at the stage of the derivation Example 7.22 represents. I maintain it here for the sake of clarity.
approach, it is possible that we have to judge a structure where no daughter percolates its features since Merge and Move do not play a role, and hence it is impossible to impose requirements on the structure via Merge and Move. Therefore it is impossible to rely on (category) features when defining \( \mathbf{X} \)-rules, as we already saw in this section.

Although we concluded that one of the reasons for requiring heads to project before they are selected as complements and specifiers is the representational nature of the formalization, it still seems advisable to obey the projection requirement in both representational and derivational approaches. The fact that Kayne's LCA requires complements to project is namely of great importance for both derivational and representational approaches, especially if we want to eliminate \( \mathbf{X} \)-Theory entirely, as Chomsky [Cho95, Page 249] proposes. If we want to eliminate \( \mathbf{X} \)-Theory we need to derive the linear order of specifiers, adjuncts, heads and complements, which is now simply given in \( \mathbf{X} \)-Theory. Since it is possible to derive this linear ordering applying Kayne's LCA, we need to obey the requirements the LCA poses on the structure. Assuming projected complements is one of those requirements.

The most important goal of the new versions of \( \mathbf{X} \)-Theory is to avoid vacuous projections. I argued that it is impossible to avoid vacuous projections since each head seems to need a complement, which causes the need for null-complements in some cases. Therefore I decided not to formalize one of the new versions of \( \mathbf{X} \)-Theory.

### 7.4 Summary

The module on \( \mathbf{X} \)-theory is identical for Chomsky's and for Zwart's framework. The latest ideas about \( \mathbf{X} \)-theory, as described by Chomsky [Cho95, Page 241ff] and Zwart [Zwa97, Page 171ff], appear not to be suitable for formalization and therefore the original \( \mathbf{X} \)-theory is applied in the formalization of Zwart's framework. The reason why the new ideas are not suitable is that specifiers, heads and complements cannot be made mutually exclusive.

Features have such a prominent role in the minimalist framework that they have to be connected with \( \mathbf{X} \)-theory. In the original \( \mathbf{X} \)-rules, [category] is the only feature that is percolated. In the formalization, I describe the percolation of category features and other features in the function \( \text{XFeatures} \). Therefore the function \( \text{XBarLevels} \), which is closest to the original \( \mathbf{X} \)-rules, does not contain category features.

The module on \( \mathbf{X} \)-theory deals with sub-categorization. The function \( \text{XRestrictions} \) defines that the specifiers and complements of a phrase must be licensed by the head of the phrase.

The positions of specifiers, heads, complements and adjuncts are explicitly defined in the \( \mathbf{X} \)-module. It is necessary to define these positions in
order to be able to use the notions ‘specifier’, ‘head’, ‘complement’ and ‘adjunct’ in the formalization. This module seems to be the correct place to represent this knowledge, because in the literature $\exists$-rules are often used to define notions such as ‘specifier’ and ‘complement’.