A Multimodal Typelogical Grammar for Verb Clusters with Past Participles in Dutch Subordinate Clauses

Davy Baardink

August 28, 2017

Contents

Abstract

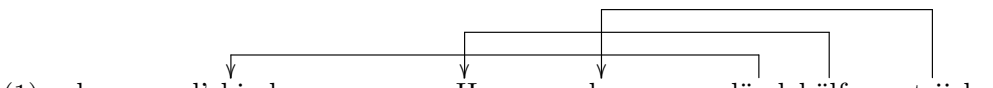
This research focuses on extending the multimodal typelogical grammar fragment for Dutch verb clusters as developed by Michael Moortgat (1999). The extended grammar fragment is intended to enable parsing of verb clusters with past participles in Dutch subordinate clauses. Based on existing theory on past participles and the multimodal typelogical grammar formalism, postulates that allow the parser to parse such clusters are added to the fragment. The extended grammar is tested against relevant subordinate clauses from the LASSY-klein corpus. Adding an additional mode to the binary operators and introducing postulates based on that mode proves to be insufficient as it allowed ungrammatical sentences to be parsed. Adding an additional mode to the unary structural operators and introducing three postulates based on this additional mode proves to be able to allow the parser to parse all sentences with one participle, but not all sentences with two participles. The three postulates are modified and three more postulates are added to the fragment for a total of six postulates that allow the parser to parse all subordinate clauses with one or more participles. The research concludes that the grammar fragment extended with the six postulates can enable parsing of verb clusters with past participles in Dutch subordinate clauses and that the research goal has been reached.

1 Introduction

In 1959 Chomsky created a hierarchy for sentence-generating grammars, where the set of Regular Grammars is a strict subset of the set of Context-Free Grammars, which is a strict subset of Context-Sensitive Grammars, which is a strict subset of Recursively Enumerable Grammars, see equation (1).

$$\begin{aligned} &\textit{Recursively Enumerable Grammars} \supset \textit{Context} - \textit{Sensitive} \\ &\qquad \textit{Grammars} \supset \textit{Context} - \textit{Free Grammars} \supset \textit{Regular Grammars} \end{aligned} \quad (1)$$

Chomsky (1956) had already proven that English could not be generated by Regular Grammars, however, he did not know whether or not all natural languages can be generated by a Context-Free Grammar.
In the 1980s, Stuart Shieber (1985) was able to prove that there are natural languages that cannot be generated by a Context-Free Grammar. More specifically, he proved that Context-Free Grammars cannot describe cross-serial dependencies in Swiss German.

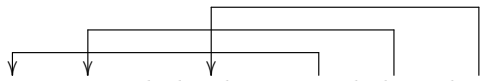


(1) das mer d'chind em Hans es huus lönd hälfe aastriiche
that we the children.ACC Hans.DAT the house.ACC let help paint

'That we let the children help Hans paint the house'

Sentence (1) gives an example of cross-serial dependencies in Swiss German, since the lines representing the dependency relations cross each other. With appropriate homomorphisms and intersection with a regular language, Swiss German can be reduced to

the copy language $\{ww|w \in \{a,b\}^*\}$, which is provably non-context free.[1] Since context-free languages are closed under homomorfims and intersection with a regular language, this means that Swiss German is also non-context free.
Earlier, Huybregts (1976) showed that cross-serial dependencies appear in Dutch as well (however, as Dutch has no case marking he was unable to proof that the string language of Dutch was context-free)[2]. Sentence (2) gives an example of cross-serial dependencies in Dutch.



(2) dat Jan Piet Marie de kinderen zag helpen leren zwemmen
that Jan Piet Marie the children saw help teach swim

'That Jan saw Piet help Marie teach the children to swim'

In both Dutch and Swiss German, these cross-serial dependencies appear because the verbs form a cluster at the end of the clause. Throughout the years, a number of different grammar formalisms have been used in order to analyse these verb clusters in Dutch, for example Transformational Grammar (Evers, 1975), Combinatory Categorial Grammar (Steedman, 1985), Minimalist Grammar (Stabler, 1997), Multimodal Typelogical Grammar (Moortgat, 1999), Lambda Grammar (Muskens, 2007) and Head-Driven Phrase Structure Grammar (Augustinus, 2015). The Multimodal Typelogical Grammar (MTLG) fragment by Michael Moortgat (1999), which will be discussed more extensive in section 2.3, is able to analyse verb clusters with infinite verbs, which only appear in descending word order. Past participles show more word order variation than infinite verbs and can therefore not be completely analyzed by Moortgat's MTLG fragment. The goal of this research is to extend that fragment so that it will be able to analyze verb clusters with past participles.

1.1 Research Question

The research question is as follows: 'How can the MTLG fragment by Michael Moortgat (1999) be extended so that it will be able to analyze verb clusters with past participles in Dutch subordinate clauses?' This research is limited to subordinate clauses because Dutch subordinate clauses are more straightforward than Dutch main clauses and because in this MTLG it is assumed that the canonical word order in Dutch appears in subordinate clauses.

1.2 Article Overview

The remainder of this article is structured as follows: Section 2.1 describes the word order variation of verb clusters with past particles as described in the literature. Section

[1] See Huybregts (1983) for the proof.
[2] See Bresnan et al, 1982

2.2 gives a summary of the workings of MTLG's. Section 2.3 describes the MTLG fragment by Michael Moortgat (1999), in particular the way it analyzes verb clusters in Dutch subordinate clauses. Section 3 presents the Case Study conducted for this research. Section 4 provides further research possibilities and concludes.

2 Theoretical Background

2.1 Word order variation of verb clusters with past participles

Verb clusters in Dutch consist of at least two verbs and theoretically have no upper bound on the number of verbs (Haeseryn, 1997). Haeseryn notes that for verb clusters with past participles and one other verb, there are two grammatical word orders: the participle can either precede (3) or follow (4) the auxiliary.

(3) Ik geloof dat hij het gedaan heeft.
I believe that he it done has

(4) Ik geloof dat hij het heeft gedaan.
I believe that he it has done.
'I believe that he has done it.'

This is supported by the data found by Augustinus in the corpora CGN and LASSY: both word orders appeared roughly equally frequently. However in written Dutch following the auxiliary was found more often while in spoken Dutch preceding the auxiliary was found more often.

	CGN		LASSY		Sum
	1-2	2-1	1-2	2-1	
$FINITE_1\ PSP_2$	1664	1626	3544	1519	8353
$INF_1\ PSP_2$	127	355	830	530	1842

Table 2.1: Two-verb clusters with Past Participles in CGN and LASSY-small.

Looking at regional spread, Broekhuis & Corver (2015) note that the order (4) is only found in the northern part of the Dutch speaking area, while it is not found in the Belgium part of the Dutch speaking area.

In clusters with three verbs participles can not only appear adjacent to their auxiliary (5) (6) but they can also be separated from their auxiliary by the third verb, in which case we get the order participle-verb-auxiliary (7) (Broekhuis & Corver, 2015).

(5) Ze denken dat hij niets kan gezien hebben
They think that he nothing can seen have

(6) Ze denken dat hij niets kan hebben gezien
They think that he nothing can have seen

(7) Ze denken dat hij niets gezien kan hebben
They think that he nothing seen can have
'They think that he couldn't have seen anything'

This is once again supported by the data found by Augustinus as three-verb clusters with past participles the past participle could appear in any position, whereas the auxiliary always has to follow the third verb.

	CGN			LASSY			Sum
	1-2-3	3-1-2	2-3-1	1-2-3	3-1-2	2-3-1	
$FINITE_1\ INF_2\ PSP_3$	89	139	63	395	262	48	996
$INF_1\ INF_2\ PSP_3$	4	18	6	33	29	4	94

Table 2.2: Three-verb clusters with Past Participles in CGN and LASSY-small.

Looking at regional spread, Haeseryn (1997) noted that in the northern variants of Dutch the order in (7) was more common, while in Belgium the order in (5) was most common. In written Dutch in both the Netherlands and Belgium the preference is the order in (6).
Clusters with more than three verbs are quite infrequent. Haeseryn (1997) notes that once again the participle can appear in every position in the cluster, however only the first (8) and last (9) positions are commonly used.

(8) De kans is groot dat de werkzaamheden gestaakt zullen moeten
The chance is big that the proceedings discontinued will have-to
worden
be

(9) De kans is groot dat de werkzaamheden zullen moeten worden
The chance is big that the proceedings will have-to be
gestaakt
discontinued

(10) De kans is groot dat de werkzaamheden zullen gestaakt moeten
The chance is big that the proceedings will discontinued have-to
worden
be

(11) De kans is groot dat de werkzaamheden zullen moeten gestaakt
The chance is big that the proceedings will have-to discontinued
worden
be
'There is a big chance that the proceedings will have to be discontinued.'

Appearing in the middle of the cluster (10) (11) is mainly used in Belgium, whereas once again having the participle as a final verb (9) is mainly used in journalistic language (Haeseryn, 1997).
Aside from the past auxiliaries 'hebben' and 'zijn', there are also passive auxiliaries ('worden' and 'krijgen'). Worden makes the direct object the subject if there is no indirect object and makes the indirect object the subject if there is one, while krijgen makes the direct object the subject if there is an indirect object. If this passive auxiliary is finite or infinite, the same word orders as in the example sentences above are possible.

However, when these passive auxiliaries appear as a participle themselves, their participle has to precede them.

(12) De Britse troepen hebben een sector toegewezen gekregen.
The Brittish troops have a sector assigned been.

(13) * De Britse troepen hebben een sector gekregen toegewezen.
The Brittish troops have a sector been assigned.
'The Brittish troops have been assigned to a sector.'

This is easy to observe for the auxiliary 'krijgen', but harder for the auxiliary 'worden'. This is because in the northern variants of Dutch the verb 'geworden' is usually deleted, so it is impossible to see if its auxiliary follows or precedes it. In Belgium, where 'geworden' is not deleted, the participle cannot follow its auxiliary anyway, according to Broekhuis & Corver (2015). However, Haeseryn (1997) notes that there are some fossilized expressions (for instance 'gespaard gebleven' in (14)) that can also appear as a past participle, in which case the main verb must precede the other participle.

(14) We zijn voor die overstroming gespaard gebleven
We are for that flood spared remained

(15) * We zijn voor die overstroming gebleven gespaard
We are for that flood remained spared
'We have been spared the flood.'

The general rule seems to be that participles have to preceed their auxiliary if the auxiliary also has the participle form. The reason why we only see this in passive participles and fossilized expressions is because other verbs either show the IPP (infinitive pro participio) effect - where if a word that should appear as a participle because it was selected by an auxiliary is itself a clustering verb it appears as an infinite verb instead of a participle - or the verb does appear as a participle but then selects its argument as a 'te infinitief'.

2.2 Multimodal Typelogical Grammars

This section is based on Moortgat (2010).

The most basic form of typelogical grammars is one that consists of atomic types and complex types that are created by the binary operations product ($A \otimes$B), left division ($A \backslash B$, pronounced A under B) and right division (A/B, pronounced A over B). When categorizing linguistic expressions, we use atomic types for grammatically 'complete' phrases, like s for declarative sentences, np for noun phrase and n for nouns. Division expresses incompleteness, so an expression with type $A \backslash B$ will produce the type B when it is put together with a phrase of type A to its left, see equation (2).

$$\frac{\dfrac{}{A \backslash B \to A \backslash B}\, id}{A \otimes A \backslash B \to B}\, \triangleleft^{-1} \tag{2}$$

Phrase B/A will produce the type B when it is put together with a phrase of type A to its right, see equation (3).

$$\frac{\dfrac{}{A/B \to A/B}\ id}{B/A \otimes A \to B}\ \triangleright^{-1} \tag{3}$$

The product operator explicitly expresses the formation of a complex phrase out of constituent parts of type A and B in that order. We can form the constituent 'Mary dreams' if 'Mary' is of the atomic type np and 'dreams' is of type $np\backslash s$, which then becomes the sentence 'Mary dreams' of type s, see equation (4).

$$\frac{np\backslash s \to np\backslash s}{np \otimes np\backslash s \to s}\ \triangleleft^{-1} \tag{4}$$

Natural Deduction is a proof calculus that allows us to calculate if a structure of words had a certain type. There are two types of structures in Natural Deduction: atomic structures and complex structures. Atomic structures are types, complex structures appear in the form $X \circ Y$, where X and Y are structures. Sequents in Natural Deduction are statements of form $X \vdash A$ (where X is a structure and A is a type) which is pronounced 'X has type A'. For the aforementioned form of typelogic grammar Natural Deduction has the axiom in (5), and introduction and elimination rules for each type forming operation as inference rules (6-11). In (11) $X[Y]$ means 'structure X with substructure Y'.

$$\frac{}{A \vdash A}\ Ax \tag{5}$$

$$\frac{A \circ X \vdash B}{X \vdash A\backslash B}\ \backslash I \tag{6}$$

$$\frac{X \vdash A \quad Y \vdash A\backslash B}{X \circ Y \vdash B}\ \backslash E \tag{7}$$

$$\frac{A \circ X \vdash B}{X \vdash A\backslash B}\ /I \tag{8}$$

$$\frac{X \vdash B/A \quad Y \vdash A}{X \circ Y \vdash B}\ /E \tag{9}$$

$$\frac{X \vdash A \quad Y \vdash B}{X \circ Y \vdash A \otimes B}\ \otimes I \tag{10}$$

$$\frac{Y \vdash A \otimes B \quad X[A \circ B] \vdash C}{X[Y] \vdash C}\ \otimes E \tag{11}$$

Natural language can be represented in Natural Deduction by representing words as atomic structures, word groups as complex structures and by having sequents state what syntactic type a sequent has. This Natural Deduction is language universal, so providing a categorical grammar for a particular language is done by specifying the lexicon of that language. The lexicon gives an atomic or complex type for each word. Therefore the grammatical notions in the language will emerge from the type structure. This will be illustrated with verb transitivity and case. Intransitive verbs, for example

'dreams', need a subject to their left to form a sentence, so in the lexicon intransitive verbs get type $np\backslash s$ (see equation 12).

$$\dfrac{\dfrac{}{Mary \vdash np}\ Ax \quad \dfrac{}{dreams \vdash np\backslash s}\ Ax}{Mary \circ dreams \vdash s}\ \backslash E \tag{12}$$

Transitive verbs, for example 'teases', require both a subject to their left and an object to their right, so in the lexicon intransitive verbs get type $(np\backslash s)/np$ (see equation 13).

$$\dfrac{\dfrac{}{Mary \vdash np}\ Ax \quad \dfrac{\dfrac{}{teases \vdash (np\backslash s)/np}\ Ax \quad \dfrac{}{Alice \vdash np}\ Ax}{teases \circ Alice \vdash np\backslash s}\ /E}{Mary \circ (teases \circ Alice) \vdash s}\ \backslash E \tag{13}$$

For case, consider pronouns in English. English pronouns have different forms depending on their case: 'he' is a subject pronoun, 'him' is an object pronoun. Subject pronouns become sentences when have an intransitive verb to their right, so they get the form $s/(np\backslash s)$ (see equation 14).

$$\dfrac{\dfrac{}{He \vdash s/(np\backslash s)}\ Ax \quad \dfrac{\dfrac{}{teases \vdash (np\backslash s)/np}\ Ax \quad \dfrac{}{Mary \vdash np}\ Ax}{teases \circ Mary \vdash np\backslash s}\ /E}{He \circ (teases \circ Mary) \vdash s}\ /E \tag{14}$$

Object pronouns become intransitive verbs when have a transitive verb to their left, so they get the form $((np\backslash s)/np)\backslash(np\backslash s)$ (see equation 15).

$$\dfrac{\dfrac{}{Mary \vdash np}\ Ax \quad \dfrac{\dfrac{}{teases \vdash (np\backslash s)/np}\ Ax \quad \dfrac{}{him \vdash ((np\backslash s)/np)\backslash(np\backslash s)}\ Ax}{teases \circ him \vdash np\backslash s}\ \backslash E}{Mary \circ (teases \circ him) \vdash s}\ \backslash E \tag{15}$$

With these types we can distinguish between the grammatical 'he likes her' and the ill-formed 'her likes he'.

A problem with this base form of typelogical grammars is that since it does not have commutativity and associativity these grammars cannot deal with word order variation. Simply introducing commutativity and asociativity to the grammar will not do, because that means the grammar will recognize any word order, while word order variation is restricted to a certain extend in most natural languages. In order to enable these word order variations we introduce multimodal typelogical grammars (MTLG), which has as a key idea that instead of a single family of type-forming operations ($\otimes$, $\backslash$, $/$), we have multiple families of operations which exist in one and the same logic. To discriminate between these families the operators get a mode index ($\otimes_i$, $\backslash_i$, $/_i$). Each family has the same logical rules but they can differ in their structural properties. In particular they can interact with in terms of structural rules that mix different modes. As an example

we take the case of the non-peripheral extractions, like 'What Mary put _ there'. The object of the verb 'put' is extracted to the WH-question 'what'. Assuming that we have 2 modes (mode c and mode d), we can introduce rules of mixed associativity (16), mixed commutativity (17) and inclusion (18).

$$mixed\ associativity:\ (A \otimes_c B) \otimes_d C \to A \otimes_c (B \otimes_d C) \tag{16}$$

$$mixed\ commutativity:\ (A \otimes_c B) \otimes_d C \to (A \otimes_d C) \otimes_c B \tag{17}$$

$$inclusion:\ A \otimes_d B \to A \otimes_c B \tag{18}$$

Then if we give 'what' (related to a non-subject np) the type $wh/_c(s/_d np)$ we can use mixed associativity, mixed commutativity and inclusion to solve this problem, see the derivation in (19)

$$\dfrac{\dfrac{}{What \vdash wh/_c(s/_d np)}\ Ax \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{(Mary \circ_c ((put \circ_c \_) \circ_c there \vdash s}}{(Mary \circ_c ((put \circ_d \_)) \circ_c there \vdash s}\ inclusion}{Mary \circ_c ((put \circ_c there) \circ_d \_) \vdash s}\ mixed\ comm.}{(Mary \circ_c (put \circ_c there)) \circ_d \_ \vdash s}\ mixed\ assoc.}{Mary \circ_c (put \circ_c there) \vdash s/_d np}\ /I}{What \circ_c (Mary \circ_c (put \circ_c there)) \vdash wh}\ /E \tag{19}$$

With these multimodal interaction principles we can avoid to overgeneralization and undergeneralization we get from global associativity and global commutativity.
Aside from modes we can also add control operators ($\Box$ and $\Diamond$) to our base logic to analyse word order variation. Unlike the previously mentioned operators, our control operators our one-place operators. With these operators, complex types can now be any of the following type: $A \otimes B$, $A \backslash B$, A/B, $\Box A$ and $\Diamond A$. Turning to Natural Deduction, complex structures can now not only appear in the form $X \circ Y$, but also in the form $\langle X \rangle$, which is the structural counterpart to $\Diamond A$. Just like the binary operators, these unary operators have introduction and elimination rules (20-23).

$$\frac{\langle X \rangle \vdash A}{X \vdash \Box A}\ \Box I \tag{20}$$

$$\frac{X \vdash \Box A}{\langle X \rangle \vdash A}\ \Box E \tag{21}$$

$$\frac{X \vdash A}{\langle X \rangle \vdash A}\ \Diamond I \tag{22}$$

$$\frac{Y \vdash \Diamond A \quad X[\langle A \rangle] \vdash B}{X[Y] \vdash B}\ \Diamond E \tag{23}$$

With these unary operators we can also produce the same mixed associativity and mixed commutativity that we showed earlier, see equation (24) and (25).

$$\Diamond\ mixed\ associativity:\ (A \otimes B) \otimes \Diamond C \to A \otimes (B \otimes \Diamond C) \tag{24}$$

$$\Diamond \textit{ mixed commutativity}: \ (A \otimes B) \otimes \Diamond C \rightarrow (A \otimes \Diamond C) \otimes B \tag{25}$$

In this case we give 'what' type $wh/(s/\Diamond\Box np)$. Once again we can show how these rules are used to prove that 'What Mary found there' is a well formed np, see equation (26).

$$\dfrac{\dfrac{}{What \vdash wh/(s/\Diamond\Box np)}\ Ax \quad \dfrac{\dfrac{\dfrac{\dfrac{\vdots}{(Mary \circ ((put \circ \_) \circ there \vdash s}\ since \Diamond\Box A \vdash A}{(Mary \circ ((put \circ \Diamond\Box\_)) \circ there \vdash s}\ \Diamond \textit{ mixed comm.}}{Mary \circ ((put \circ there) \circ \Diamond\Box\_) \vdash s}\ \Diamond \textit{ mixed assoc.}}{(Mary \circ (put \circ there)) \circ \Diamond\Box\_ \vdash s}\ /I}{Mary \circ (put \circ there) \vdash s/\Diamond\Box np}\ /E}{What \circ (Mary \circ (put \circ there)) \vdash wh} \tag{26}$$

Aside from licensing structural transformation, as we did just yet, we can also use the unary operators to block structural transformation. This can be shown by looking at extraction islands, phrases that do not allow a gap to be left behind. For example, the noun phrase 'the concert' in the sentence 'Mary fell asleep during the concert' cannot be moved because it is part of the adjunct 'during the concert'. In order to make this move impossible we can assign during the type $(\Box(s\backslash s))/np$. In order to eliminate the box we first need to form $\Box(s\backslash s)$ by having an np to the right. Therefore we cannot move the np, because if we do, we will be unable to unbox the $s\backslash s$.

2.3 An MTLG fragment for analyzing verb clusters

This MTLG fragment assumes that the SOV word order of Dutch subordinate clauses is canonical and that verbs canonically select their complements to their left, which is based on Koster (1974, as cited by Moortgat, 1999). Transitive verbs therefore get type $np\backslash(np\backslash s)$ as opposed to the type $(np\backslash s)/np$ that we used in the previous section. Current teaching has it that the English SVO order is canonical, but "for this to work, you need to introduce a generous supply of abstract syntactic positions for complements to move to" (Moortgat, 1999), and typelogical grammars have no syntactic positions. In order to analyse verb clusters in Dutch subordinate clauses Moortgat's MTLG fragment (1999) uses both modes and the unary structural operators. The fragment uses different modes for sequent types and for structures. The only mode for sequent types that is considered in this paper is mode e, for subordinate clauses (e stands for verb at the End). For structures, the fragment uses mode 0 for verbal material in the structure and mode 1 for non-verbal material in the structure. Every verbal head in these structures gets a $\Box_0$, since they are verbal material. We then need postulates to make sure that the $\Box_e$, from the clause type, will translate to the $\Box_0$'s on the verbal heads. These $\Box_0$'s act like locks that need to be unlocked in order to analyse these sentences. In order to unlock these locks we require the corresponding $\Diamond_0$ 'key'. We require postulates to obtain the $\Diamond_0$ key from the $\Diamond_e$ that we obtain from the clause type. In order to get from the $\Box_e$ on the goal formula to the lexical $\Box_0$ we start with postulates P1 (27) and P2 (28).

$$P1 : \Diamond_e(A \otimes_1 B) \rightarrow A \otimes_1 \Diamond_e B \tag{27}$$

$$P2 : \Diamond_e A \to \Diamond_0 A \tag{28}$$

Postulate P1 allows the $\Diamond_e$, which we get from the clause type, to recursively traverse the phrasal structure, looking for the verbal head on the right branch of the tree. At the end of the recursion, P2 switches the $\Diamond_e$ to a $\Diamond_0$, see equation (29) for an example.

$$\dfrac{\dfrac{\overline{dat \vdash dat/_1\Box_e s}\ Ax \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Alice \vdash np}\ Ax \quad \dfrac{\dfrac{}{slaapt \vdash \Box_0(np\backslash_1 s)}\ Ax}{\langle slaapt\rangle^0 \vdash np\backslash_1 s}\ \Box E}{Alice \circ_1 \langle slaapt\rangle^0 \vdash s}\ \backslash E}{Alice \circ_1 \langle slaapt\rangle^e \vdash s}\ P2}{\langle Alice \circ_1 slaapt\rangle^e \vdash s}\ P1}{Alice \circ_1 slaapt \vdash \Box_e s}\ \Box I}{dat \circ_1 (Alice \circ_1 slaapt) \vdash dat}\ /E \tag{29}$$

'that Alice sleeps.'

The only verb clusters that this version of this MTLG looks at are verb clusters with modal or perception verbs, both of which select bare infinitives as their complements. These verbs combine with the verbal head of their complement, and pass over the non-verbal part of their complement, which creates the crossing dependencies that have been discussed in the introduction, see equation (30) for an example.

$$\dfrac{\dfrac{\vdots}{\langle wil \circ_0 ((de \circ_1 soepschildpad) \circ_1 plagen)\rangle^0 \vdash np\backslash_1 s}\ ???}{\vdots} \tag{30}$$

'wants to tease the mock turtle.'

In order to make this work we need a postulate that allows other structures to skip over verbs. The postulate † (31), allows structures to skip over other structures.

$$\dagger : A \otimes (B \otimes C) \to B \otimes (A \otimes C) \tag{31}$$

However, postulate † would mean that structures would be able to skip over any kind of words, not just verbs.

$$\dfrac{\dfrac{\vdots}{hij \circ_1 ((de \circ_1 soepschildpad) \circ_1 \langle wil \circ_0 plagen\rangle^0) \vdash s}\ \dagger}{\vdots} \tag{32}$$

'the mock turtle wants to tease he.' (subordinate clause)

We need to strictly define that this move is only possible for words that combine with operators of mode 0, in other words the verbal part of the clause. This modification of our postulate is described in postulate ‡ (33).

$$‡ : A \otimes_1 (B \otimes_0 C) \rightarrow B \otimes_0 (A \otimes_1 C) \tag{33}$$

However, with this postulate the English SVO order, which is ungrammatical in Dutch, would still be accepted, because it is actually the premise for the ‡ step, see equation (34).

$$\dfrac{\dfrac{\vdots}{\langle wil \circ_0 ((de \circ_1 soepschildpad) \circ_1 plagen)\rangle^0 \vdash np\backslash_1 s}}{\dfrac{(de \circ_1 soepschildpad) \circ_1 \langle wil \circ_0 plagen\rangle^0 \vdash np\backslash_1 s}{\vdots}}\ ‡ \tag{34}$$

'wants to tease the mock turtle.'

In order to accommodate for that, we turn back to our $\diamondsuit_0$, which we will first distribute over the $\circ_0$, with postulate P3 (35), which checks if the components of the cluster are both verbal heads.

$$P3 : \diamondsuit_0(A \otimes_0 B) \rightarrow \diamondsuit_0 A \otimes_0 \diamondsuit_0 B \tag{35}$$

Afterwards, we can change postulate ‡ into postulate P4 (36), which has the inference that the word to be moved in the tree is required to have a $\diamondsuit_0$.

$$P4 : A \circ_1 (\diamondsuit_0 B \otimes_0 C) \rightarrow \diamondsuit_0 B \otimes_0 (A \otimes_1 C) \tag{36}$$

Therefore, that word has to have been moved before we can assign its lexical type, as its lock still needs to be unlocked by the $\diamondsuit_0$, but before it can be unlocked it needs to have moved, see the derivation in (37).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{dat \vdash dat/_1\Box_e s}\ Ax \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Alice \vdash np}\ Ax \qquad \dfrac{\dfrac{}{wil \vdash \Box_0(inf\backslash_0((np\backslash_1 s))}\ Ax}{\langle wil\rangle^0 \vdash (np\backslash_1 s)/_0 inf}\ \Box E \qquad \dfrac{\dfrac{\dfrac{}{de \vdash np/_1 n}\ Ax \qquad \dfrac{}{soepschildpad \vdash n}\ Ax}{de \circ_1 soepschildpad \vdash np}\ /E \qquad \dfrac{\dfrac{}{plagen \vdash \Box_0(np\backslash_1 inf)}\ Ax}{\langle plagen\rangle^0 \vdash np\backslash_1 inf}\ \Box E}{(de \circ_1 soepschildpad) \circ_1 \langle plagen\rangle^0 \vdash inf}\ \backslash E}{\langle wil\rangle^0 \circ_0 ((de \circ_1 soepschildpad) \circ_1 \langle plagen\rangle^0) \vdash np\backslash_1 s}\ /E}{Alice \circ_1 \circ_1(\langle wil\rangle^0 \circ_0 ((de \circ_1 soepschildpad) \circ_1 \langle plagen\rangle^0)) \vdash s}\ \backslash E}{Alice \circ_1 (de \circ_1 soepschildpad) \circ_1 (\langle wil\rangle^0 \circ_0 \langle plagen\rangle^0) \vdash s}\ P4}{Alice \circ_1 (de \circ_1 soepschildpad) \circ_1 \langle wil \circ_0 plagen\rangle^0 \vdash s}\ P3}{Alice \circ_1 (de \circ_1 soepschildpad) \circ_1 \langle wil \circ_0 plagen\rangle^e \vdash s}\ P2}{Alice \circ_1 \langle (de \circ_1 soepschildpad) \circ_1 (wil \circ_0 plagen)\rangle^e \vdash s}\ P1}{\langle Alice \circ_1 ((de \circ_1 soepschildpad) \circ_1 (wil \circ_0 plagen))\rangle^e \vdash s}\ P1}{Alice \circ_1 ((de \circ_1 soepschildpad) \circ_1 (wil \circ_0 plagen)) \vdash \Box_e s}\ \Box I}{dat \circ_1 (Alice \circ_1 ((de \circ_1 soepschildpad) \circ_1 (wil \circ_0 plagen))) \vdash dat}\ /E \tag{37}$$

'that Alice wants to tease the mock turtle.'

Alternatively we can replace P3 and P4 by the postulates P3 and P4 in (38) and (39).

$$P3 : \Diamond_0(A \otimes_0 B) \rightarrow \Diamond_0 B \otimes_0 \Diamond_0 A \tag{38}$$

$$P4 : A \circ_1 (B \otimes_0 \Diamond_0 C) \rightarrow (A \otimes_1 B) \otimes_0 \Diamond_0 C \tag{39}$$

These postulates yield the same acceptance of sentences but differ in that verbs that take verbs as complements have these complements appear to their left rather than to their right. In other words, they get type $\Box_0(inf \backslash_0 (np \backslash_1 s))$ rather than type $\Box_0((np \backslash_1 s)/_0 inf)$. See equation 40 for an example.

$$\dfrac{\dfrac{\overline{Alice \vdash np}\ Ax \qquad \dfrac{\dfrac{\dfrac{\overline{de \vdash np/_1 n}\ Ax \quad \overline{soepschildpad \vdash n}\ Ax}{de \circ_1 soepschildpad \vdash np}\ /E \quad \dfrac{\overline{plagen \vdash \Box_0(np \backslash_1 inf)}\ Ax}{\langle plagen \rangle^0 \vdash np \backslash_1 inf}\ \Box E}{(de \circ_1 soepschildpad) \circ_1 \langle plagen \rangle^0 \vdash inf}\ \backslash E \quad \dfrac{\overline{wil \vdash \Box_0(inf \backslash_0 (np \backslash_1 s))}\ Ax}{\langle wil \rangle^0 \vdash inf \backslash_0 (np \backslash_1 s)}\ \Box E}{((de \circ_1 soepschildpad) \circ_1 \langle plagen \rangle^0) \circ_0 \langle wil \rangle^0 \vdash np \backslash_1 s}\ \backslash E}{Alice \circ_1 \circ_1 (((de \circ_1 soepschildpad) \circ_1 \langle plagen \rangle^0) \circ_0 \langle wil \rangle^0) \vdash s}\ P4}{Alice \circ_1 (de \circ_1 soepschildpad) \circ_1 (\langle plagen \rangle^0 \circ_0 \langle wil \rangle^0) \vdash s}\ P3}{Alice \circ_1 (de \circ_1 soepschildpad) \circ_1 \langle wil \circ_0 plagen \rangle^0 \vdash s}\ P2}{Alice \circ_1 (de \circ_1 soepschildpad) \circ_1 \langle wil \circ_0 plagen \rangle^e \vdash s}\ P1}{Alice \circ_1 \langle (de \circ_1 soepschildpad) \circ_1 (wil \circ_0 plagen) \rangle^e \vdash s}\ P1}{\langle Alice \circ_1 ((de \circ_1 soepschildpad) \circ_1 (wil \circ_0 plagen)) \rangle^e \vdash s}\ \Box I}{Alice \circ_1 ((de \circ_1 soepschildpad) \circ_1 (wil \circ_0 plagen)) \vdash \Box_e s}\ /E}{dat \circ_1 (Alice \circ_1 ((de \circ_1 soepschildpad) \circ_1 (wil \circ_0 plagen))) \vdash dat} \tag{40}$$

$$\overline{dat \vdash dat/_1 \Box_e s}\ Ax$$

'that Alice wants to tease the mock turtle.'

The advantage of these postulates over the previous ones is that it conforms to the generalization that Dutch verbs select their complements uniformly to the left. This version of P3 and P4 is the version that will be used in the fragment described in the case study.

3 Case Study

3.1 Clusters with one participle

Te example sentences in this section were obtained from the LASSY-small corpus. First, lets recap the postulates from the fragment by Moortgat (1999).

$$P1 : \Diamond_e(A \otimes_1 B) \rightarrow A \otimes_1 \Diamond_e B \tag{41}$$

$$P2 : \Diamond_e A \rightarrow \Diamond_0 A \tag{42}$$

$$P3 : \Diamond_0(A \otimes_0 B) \rightarrow \Diamond_0 B \otimes_0 \Diamond_0 A \tag{43}$$

$$P4 : A \circ_1 (B \otimes_0 \Diamond_0 C) \rightarrow (A \otimes_1 B) \otimes_0 \Diamond_0 C \tag{44}$$

As past participles differ in terms of their structural variation compared to finite and infinite verbs, it is necessary to introduce an additional mode for past participles. This

additional mode, which will be called p, can either be applied to the binary operators that take both the participle and its selector as arguments, or to the unary operators that take the participle as their argument. Let us first consider applying this mode to the binary operators. For clusters with two verbs, the participle can either appear to the left (16) or to the right (17) of its selector.

(16) Als u uw reisdocument bent verloren ...
If you your travel-document are lost ...
'If you have lost your traveldocument ...' [WR-P-P-C-0000000011.p.23.s.1]

(17) ... of u de medicijnen voor die dag vergeten bent.
... if you the medicines for that day forgotten are.
'... if you have forgotten your medicines for that day.' [WR-P-P-C-0000000054.txt-275.2]

If the participle appears to the right of its selector, it behaves in the same way as the infinitival complements to modal and perception verbs. Therefore, we introduce a modified version of P3 that works with participle branches, which will be called PP3 (PP stands for participle postulate) (45). See equation (46) for a the relevant part of the derivation of sentence (16).

$$PP3 \Diamond_0(A \otimes_p B) \rightarrow \Diamond_0 B \otimes_p \Diamond_0 A \tag{45}$$

$$\dfrac{\dfrac{\dfrac{\vdots}{u \circ_1 ((uw \circ_1 reisdocument) \circ_1 (\langle verloren \rangle^0 \circ_p \langle bent \rangle^0)) \vdash s}\, PP3}{u \circ_1 ((uw \circ_1 reisdocument) \circ_1 \langle bent \circ_p verloren \rangle^0) \vdash s}\, P2}{u \circ_1 ((uw \circ_1 reisdocument) \circ_1 \langle bent \circ_p verloren \rangle^e) \vdash s}\, P1 \quad \vdots \tag{46}$$

If the participle appears to the left of its selector, the order of the verbs is reversed by means of postulate PP31 (47), so postulate PP3 can work with it. See equation (48) for a the relevant part of the derivation of sentence (17).

$$PP31 : \Diamond_0(A \otimes_p B) \rightarrow \Diamond_0(B \otimes_p A) \tag{47}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{u \circ_1 (((de \circ_1 medicijnen) \circ_1 (voor \circ_1 (die \circ_1 dag))) \circ_1 (\langle vergeten \rangle^0 \circ_p \langle bent \rangle^0)) \vdash s}\, PP3}{u \circ_1 (((de \circ_1 medicijnen) \circ_1 (voor \circ_1 (die \circ_1 dag))) \circ_1 \langle bent \circ_p vergeten \rangle^0) \vdash s}\, PP31}{u \circ_1 (((de \circ_1 medicijnen) \circ_1 (voor \circ_1 (die \circ_1 dag))) \circ_1 \langle vergeten \circ_p bent \rangle^0) \vdash s}\, P2}{u \circ_1 (((de \circ_1 medicijnen) \circ_1 (voor \circ_1 (die \circ_1 dag))) \circ_1 \langle vergeten \circ_p bent \rangle^e) \vdash s}\, P1 \quad \vdots \tag{48}$$

With these two postulates both word orders in clusters with two verbs will be accepted. In verb clusters with three verbs, the participle can either appear to the right of its

selector (18), to the left of its selector (19) or separated from its selector by the third verb (20).

(18) ... dat Che zou zijn omgekomen
... that Che should have died
'... that Che presumably had died.' [WR-P-E-I-0000004745.p.5.s.58]

(19) ... dat de diensten zullen uitgevoerd zijn.
... that the services will carried-out be.
'... that the services will be carried out.' [dpc-fsz-000543-nl-sen.p.201.s.1]

(20) ... dat kinderen behandeld moeten worden.
... that children treated have-to be.
'... that children have to be treated.' [WS-U-E-A-0000000040.p.18.s.2]

Postulates PP3 and PP31 are sufficient to deal with the orders in (18) and (19), see equations (49) and (50) respectively.

$$\dfrac{\dfrac{\dfrac{\vdots}{che \circ_1 ((\langle omgekomen\rangle^0 \circ_p \langle zijn\rangle^0) \circ_0 \langle zou\rangle^0) \vdash s}\, PP3}{che \circ_1 (\langle zijn \circ_p omgekomen\rangle^0 \circ_0 \langle zou\rangle^0) \vdash s}\, P3}{che \circ_1 \langle zou \circ_0 (zijn \circ_p omgekomen)\rangle^0 \vdash s}\, P2 \qquad (49)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{(de \circ_1 diensten) \circ_1 ((\langle uitgevoerd\rangle^0 \circ_p \langle zijn\rangle^0) \circ_0 \langle zullen\rangle^0) \vdash s}\, PP3}{(de \circ_1 diensten) \circ_1 (\langle zijn \circ_p uitgevoerd\rangle^0 \circ_0 \langle zullen\rangle^0) \vdash s}\, PP31}{(de \circ_1 diensten) \circ_1 (\langle uitgevoerd \circ_p zijn\rangle^0 \circ_0 \langle zullen\rangle^0) \vdash s}\, P3}{(de \circ_1 diensten) \circ_1 \langle zullen \circ_0 (uitgevoerd \circ_p zijn)\rangle^0) \vdash s}\, P2 \qquad (50)$$

However, in order to deal with the order where the participle is separated from its selector, we need to introduce a third postulate that allows the participle to pass over the third verb. Postulate PP32 (51) does exactly that. See equation (52) for the relevant part of the derivation of sentence (20)

$$PP32 : \Diamond_0 A \otimes_p (B \circ_0 C) \rightarrow \Diamond_0 B \otimes_0 (A \circ_p C) \qquad (51)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{kinderen \circ_1 (\langle behandeld \rangle^0 \circ_p \langle worden \rangle^0 \circ_0 \langle moeten \rangle^0) \vdash s}}{kinderen \circ_1 (\langle worden \circ_p behandeld \rangle^0 \circ_0 \langle moeten \rangle^0) \vdash s}\, PP3}{kinderen \circ_1 (\langle behandeld \circ_p worden \rangle^0 \circ_0 \langle moeten \rangle^0) \vdash s}\, PP31}{kinderen \circ_1 \langle moeten \circ_0 (behandeld \circ_p worden) \rangle^0) \vdash s}\, P3}{kinderen \circ_1 \langle behandeld \circ_p (moeten \circ_0 worden) \rangle^0) \vdash s}\, PP32}{kinderen \circ_1 \langle behandeld \circ_p (moeten \circ_0 worden) \rangle^e) \vdash s}\, P2}{\vdots}\, P1 \tag{52}$$

Finally, the participles can themselves select an object, just like other main verbs. Postulate P4 allows objects to be separated from their selectors if these selectors are part of the verbal component (mode 0). We introduce postulate PP4, which is a modified version of postulate P4 but works on mode p rather than on mode 0 (53). See equation (54) for the relevant part of the derivation of sentence (16).

$$PP4 : (A \otimes_1 B \otimes_p) \diamondsuit_0 C \rightarrow A \otimes_1 (B \otimes_0 \diamondsuit_0 C) \tag{53}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{u \circ_1 (((uw \circ_1 reisdocument) \circ_1 \langle verloren \rangle^0) \circ_p \langle bent \rangle^0) \vdash s}}{u \circ_1 ((uw \circ_1 reisdocument) \circ_1 (\langle verloren \rangle^0 \circ_p \langle bent \rangle^0)) \vdash s}\, PP4}{u \circ_1 ((uw \circ_1 reisdocument) \circ_1 \langle bent \circ_p verloren \rangle^0) \vdash s}\, PP3}{u \circ_1 ((uw \circ_1 reisdocument) \circ_1 \langle bent \circ_p verloren \rangle^e) \vdash s}\, P2}{\vdots}\, P1 \tag{54}$$

For clusters with more than three verbs, for example sentence (21), these postulates allow the participle to appear in every position, see equation (55) for a the relevant part of the derivation of sentence (21).

(21) ... dat beide schouders bedekt zouden moeten zijn.
... that both shoulders covered should have-to be.
'... that both shoulders would have to be covered.' [WR-P-E-I-0000022743.p.4.s.119]

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\vdots
}{(beide \circ_1 schouders) \circ_1 ((((\langle bedekt \rangle^0 \circ_p \langle zijn \rangle^0) \circ_0 \langle moeten \rangle^0) \circ_0 \langle zouden \rangle^0) \vdash s}\, PP3
}{(beide \circ_1 schouders) \circ_1 ((\langle zijn \circ_p bedekt \rangle^0 \circ_0 \langle moeten \rangle^0) \circ_0 \langle zouden \rangle^0) \vdash s}\, PP31
}{(beide \circ_1 schouders) \circ_1 ((\langle bedekt \circ_p zijn \rangle^0 \circ_0 \langle moeten \rangle^0) \circ_0 \langle zouden \rangle^0) \vdash s}\, P3
}{(beide \circ_1 schouders) \circ_1 (\langle (moeten \circ_0 (bedekt \circ_p zijn)) \rangle^0) \circ_0 \langle zouden \rangle^0) \vdash s}\, PP32
}{(beide \circ_1 schouders) \circ_1 (\langle (bedekt \circ_p (moeten \circ_0 zijn)) \rangle^0) \circ_0 \langle zouden \rangle^0) \vdash s}\, P3
}{(beide \circ_1 schouders) \circ_1 \langle zouden \circ_0 (bedekt \circ_p (moeten \circ_0 zijn)) \rangle^0) \vdash s}\, PP32
}{(beide \circ_1 schouders) \circ_1 \langle bedekt \circ_p (zouden \circ_0 (moeten \circ_0 zijn)) \rangle^0) \vdash s}\, P2
}{(beide \circ_1 schouders) \circ_1 \langle bedekt \circ_p (zouden \circ_0 (moeten \circ_0 zijn)) \rangle^e) \vdash s}\, P1
$$

$$
\vdots
\tag{55}
$$

Unfortunately, this method of applying the mode to the binary operator runs into a problem: since the operator takes both the participle and the auxiliary as arguments, the postulates can move the auxiliary the same way they move the participle. This leads to the acceptance of ungrammatical sentences, see equation 56.

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\vdots
}{kinderen \circ_1 (\langle behandeld \rangle^0 \circ_p \langle worden \rangle^0 \circ_0 \langle moeten \rangle^0) \vdash s}\, PP3
}{kinderen \circ_1 (\langle worden \circ_p behandeld \rangle^0 \circ_0 \langle moeten \rangle^0) \vdash s}\, P3
}{kinderen \circ_1 \langle moeten \circ_0 (worden \circ_p behandeld) \rangle^0) \vdash s}\, PP32
}{kinderen \circ_1 \langle worden \circ_p (moeten \circ_0 behandeld) \rangle^0) \vdash s}\, P2
}{kinderen \circ_1 \langle worden \circ_p (moeten \circ_0 behandeld) \rangle^e) \vdash s}\, P1
$$

$$
\vdots
\tag{56}
$$

The only way for the postulates to recognize which argument of an indexed binary operator needs to be moves, without using indexed unary operators, is by looking at the order of the arguments. Differentiating the arguments of a binary operator by looking at their order would be possible in this grammar if it weren't for postulates PP3 and PP31 switching the order of the arguments. Since it has to be possible to change the word order in order to accept all grammatical word order variations of participles in Dutch verb clusters, it is impossible to differentiate between the arguments of an indexed binary operator, therefore applying this additional mode to the binary operator is not going to be sufficient for analyzing Dutch verb clusters with participles.

So we turn to applying the mode to the unary operators instead. We will type participles $\square_p pastpart$, finite auxiliaries $\square_0(pastpart \backslash_0 (np \backslash_0 s))$ and infinite auxiliaries $\square_0(pastpart \backslash_0 inf)$. Considering clusters with two verbs, the participle can either appear to the left (16) or to the right (17) of its selector. If the participle appears to the right of its selector, it behaves the same as infinite verbs. Therefore, we introduce PP1 that turns $\diamondsuit_p$ into $\diamondsuit_0$ (57) after which P3 can reverse the order of the verbs. See equation

(58) for a the relevant part of the derivation of sentence (16).

$$PP1 : \Diamond_p A \to \Diamond_0 A \tag{57}$$

$$\begin{array}{l} \vdots \\ \dfrac{\dfrac{\dfrac{\dfrac{u \circ_1 ((uw \circ_1 reisdocument) \circ_1 (\langle verloren \rangle^p \circ_0 \langle bent \rangle^0)) \vdash s}{u \circ_1 ((uw \circ_1 reisdocument) \circ_1 (\langle verloren \rangle^0 \circ_0 \langle bent \rangle^0)) \vdash s}\, PP1}{u \circ_1 ((uw \circ_1 reisdocument) \circ_1 \langle bent \circ_0 verloren \rangle^0) \vdash s}\, P3}{u \circ_1 ((uw \circ_1 reisdocument) \circ_1 \langle bent \circ_0 verloren \rangle^e) \vdash s}\, P2}{\vdots}\, P1 \end{array} \tag{58}$$

If the participle appears to the right of its selector, it needs to be unaffected by P3. Instead, we introduce PP2 (59) that when the $\Diamond_0$ key reaches the participle, it divides into a $\Diamond_p$ key for the participle and a $\Diamond_0$ key for the auxiliary, without reversing their orders. See equation (60) for a the relevant part of the derivation of sentence (17).

$$PP2 : \Diamond_0 (A \otimes_0 B) \to \Diamond_p A \otimes_0 \Diamond_0 B \tag{59}$$

$$\begin{array}{l} \vdots \\ \dfrac{\dfrac{\dfrac{u \circ_1 (((de \circ_1 medicijnen) \circ_1 (voor \circ_1 (die \circ_1 dag))) \circ_1 (\langle vergeten \rangle^p \circ_0 \langle bent \rangle^0)) \vdash s}{u \circ_1 (((de \circ_1 medicijnen) \circ_1 (voor \circ_1 (die \circ_1 dag))) \circ_1 \langle vergeten \circ_0 bent \rangle^0) \vdash s}\, PP2}{u \circ_1 (((de \circ_1 medicijnen) \circ_1 (voor \circ_1 (die \circ_1 dag))) \circ_1 \langle vergeten \circ_0 bent \rangle^e) \vdash s}\, P2}{\vdots}\, P1 \end{array} \tag{60}$$

With these two participles, sentences (18) and (19) can also be parsed, see equations (61) and (62) respectively.

$$\begin{array}{l} \vdots \\ \dfrac{\dfrac{\dfrac{\dfrac{che \circ_1 ((\langle omgekomen \rangle^p \circ_0 \langle zijn \rangle)^0 \circ_0 \langle zou \rangle^0) \vdash s}{che \circ_1 ((\langle omgekomen \rangle^0 \circ_0 \langle zijn \rangle)^0 \circ_0 \langle zou \rangle^0) \vdash s}\, PP1}{che \circ_1 (\langle zijn \circ_0 omgekomen \rangle^0 \circ_0 \langle zou \rangle^0) \vdash s}\, P3}{che \circ_1 \langle zou \circ_0 (zijn \circ_0 omgekomen) \rangle^0 \vdash s}\, P3}{\vdots}\, P2 \end{array} \tag{61}$$

$$\begin{array}{l} \vdots \\ \dfrac{\dfrac{\dfrac{(de \circ_1 diensten) \circ_1 ((\langle uitgevoerd \rangle^p \circ_0 \langle zijn \rangle^0) \circ_0 \langle zullen \rangle^0) \vdash s}{(de \circ_1 diensten) \circ_1 (\langle uitgevoerd \circ_0 zijn \rangle^0 \circ_0 \langle zullen \rangle^0) \vdash s}\, PP2}{(de \circ_1 diensten) \circ_1 \langle zullen \circ_0 (uitgevoerd \circ_0 zijn) \rangle^0) \vdash s}\, P3}{\vdots}\, P2 \end{array} \tag{62}$$

Again, it is the sentence order in (20) that requires an additional postulate. Thanks to PP2, the participle will end up at the leftmost part of the verb cluster, which is its

canonical position, even in larger verb clusters. However, in order to be selected by its selector, the participle needs to be a sister node to its selector. In order to ensure that, we need to introduce associativity for partciiples only. This associativity is captured in postulate PP3 (63). See equations (64) and (65) for the relevant part of the derivations of sentences (20) and (21) respectively.

$$PP3 : \Diamond_p A \otimes_0 (\Diamond_0 B \otimes_0 \Diamond_0 C) \rightarrow (\Diamond_p A \otimes_0 \Diamond_0 B) \otimes_0 \Diamond_0 C \tag{63}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{kinderen \circ_1 ((\langle behandeld \rangle^p \circ_0 \langle worden \rangle^0) \circ_0 \langle moeten \rangle^0) \vdash s}\, PP3}{kinderen \circ_1 (\langle behandeld \rangle^p \circ_0 (\langle worden \rangle^0 \circ_0 \langle moeten \rangle^0)) \vdash s}\, P3}{kinderen \circ_1 (\langle behandeld \rangle^p \circ_0 \langle moeten \circ_0 worden \rangle^0) \vdash s}\, PP2}{kinderen \circ_1 \langle behandeld \circ_0 (moeten \circ_0 worden) \rangle^0) \vdash s}\, P2}{kinderen \circ_1 \langle behandeld \circ_0 (moeten \circ_0 worden) \rangle^e) \vdash s}\, P1}{\vdots} \tag{64}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{(beide \circ_1 schouders) \circ_1 (((\langle bedekt \rangle^p \circ_0 \langle zijn \rangle^0) \circ_0 \langle moeten \rangle^0) \circ_0 \langle zouden \rangle^0) \vdash s}\, PP3}{(beide \circ_1 schouders) \circ_1 ((\langle bedekt \rangle^p \circ_0 (\langle zijn \rangle^0 \circ_0 \langle moeten \rangle^0)) \circ_0 \langle zouden \rangle^0) \vdash s}\, PP3}{(beide \circ_1 schouders) \circ_1 (\langle bedekt \rangle^p \circ_0 ((\langle zijn \rangle^0 \circ_0 \langle moeten \rangle^0) \circ_0 \langle zouden \rangle^0)) \vdash s}\, P3}{(beide \circ_1 schouders) \circ_1 (\langle bedekt \rangle^p \circ_0 (\langle (moeten \circ_0 zijn) \rangle^0 \circ_0 \langle zouden \rangle^0)) \vdash s}\, P3}{(beide \circ_1 schouders) \circ_1 (\langle bedekt \rangle^p \circ_0 \langle zouden \circ_0 (moeten \circ_0 zijn) \rangle^0) \vdash s}\, PP2}{(beide \circ_1 schouders) \circ_1 \langle bedekt \circ_0 (zouden \circ_0 (moeten \circ_0 zijn) \rangle^0) \vdash s}\, P2}{(beide \circ_1 schouders) \circ_1 \langle bedekt \circ_0 (zouden \circ_0 (moeten \circ_0 zijn) \rangle^e) \vdash s}\, P1}{\vdots} \tag{65}$$

Finally, since P4 only requires its final argument to have a $\Diamond_0$, that postulate is sufficient to get the object to its selector, even if the selector is a participle, see equation (66).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{u \circ_1 (((uw \circ_1 reisdocument) \circ_1 (\langle verloren \rangle^p \circ_0 \langle bent \rangle^0) \vdash s}\, P4}{u \circ_1 ((uw \circ_1 reisdocument) \circ_1 (\langle verloren \rangle^p \circ_0 \langle bent \rangle^0)) \vdash s}\, PP1}{u \circ_1 ((uw \circ_1 reisdocument) \circ_1 (\langle verloren \rangle^0 \circ_0 \langle bent \rangle^0)) \vdash s}\, P3}{u \circ_1 ((uw \circ_1 reisdocument) \circ_1 \langle bent \circ_0 verloren \rangle^0) \vdash s}\, P2}{u \circ_1 ((uw \circ_1 reisdocument) \circ_1 \langle bent \circ_0 verloren \rangle^e) \vdash s}\, P1}{\vdots} \tag{66}$$

With these three postulates, the grammar accepts clusters that vary in the position of the participle, but rejects clusters where the other verbs appear in any order other than the grammatical one.

3.2 Clusters with two participles

Clusters with two participles are rare. Since there were no sentences with two participles in the same cluster in the LASSY-small corpus, the example sentences in this section were found on Google.nl on August 15, 2017. For clusters with two participles and one other verb, where one of the participles is selected by the other there are three possible word orders. The participles can appear together at the beginning of the cluster (22), together at the end of the cluster (23), or separated by the third verb (24).

(22) ... dat hij een ipadres toegewezen gekregen heeft.
... that he an ip-adress assigned been has.
'... that he has been assigned to an ip adress.'

(23) ... dat ik sinds kort een ander ipadres heb toegewezen gekregen.
... that I since shortly a different ip-adres have assigned been.
'... that I recently have been assigned to a different ip adres.'

(24) ... of ik een woning toegewezen heb gekregen.
... if I a residence assigned have been.
'... if I have been assigned to a residence.'

With the three postulates we have so far, only the order in (23) can be accepted, see equation (67).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle toegewezen \rangle^p) \circ_0 \langle gekregen \rangle^p) \circ_0 \langle heb \rangle^0)) \vdash s}\, PP1}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle toegewezen \rangle^p) \circ_0 \langle gekregen \rangle^0) \circ_0 \langle heb \rangle^0)) \vdash s}\, P4}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 (((een \circ_1 (ander \circ_1 ipadres)) \circ_1 (\langle toegewezen \rangle^p \circ_0 \langle gekregen \rangle^0)) \circ_0 \langle heb \rangle^0)) \vdash s}\, PP2}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 (((een \circ_1 (ander \circ_1 ipadres)) \circ_1 (\langle toegewezen \circ_0 gekregen \rangle^0)) \circ_0 \langle heb \rangle^0)) \vdash s}\, P4}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een(ander \circ_1 \circ_1 ipadres)) \circ_1 (\langle toegewezen \circ_0 gekregen \rangle^0 \circ_0 \langle heb \rangle^0))) \vdash s}\, P3}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle heb \circ_0 (toegewezen \circ_0 gekregen) \rangle^0)) \vdash s}\, P2$$

$$\vdots \tag{67}$$

The reason why the other two orders aren't accepted is because our postulate PP3 is too restrictive. It requires two out of the three verbs to have a $\diamondsuit_0$ key, while in the sentences considered in this section, that is not the case. Therefore we introduce a less strict form of PP3, which we will call PP3' (68).

$$PP3' : \diamondsuit_p A \otimes_0 (B \otimes_0 C) \to (\diamondsuit_p A \otimes_1 B) \otimes_0 C \tag{68}$$

Thanks to PP3', the word order in the verb clusters of sentences 22 and 24 can be accepted, see the derivation in equations (69) and (70) respectively.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{hij \circ_1 (((een \circ_1 ipadres) \circ_1 (\langle toegewezen\rangle^p \circ_0 \langle gekregen\rangle^p)) \circ_0 \langle heeft\rangle^0) \vdash s}\,P4}{hij \circ_1 ((een \circ_1 ipadres) \circ_1 ((\langle toegewezen\rangle^p \circ_0 \langle gekregen\rangle^p) \circ_0 \langle heeft\rangle^0)) \vdash s}\,PP3'}{hij \circ_1 ((een \circ_1 ipadres) \circ_1 (\langle toegewezen\rangle^p \circ_0 (\langle gekregen\rangle^p \circ_0 \langle heeft\rangle^0))) \vdash s}\,PP2}{hij \circ_1 ((een \circ_1 ipadres) \circ_1 (\langle toegewezen\rangle^p \circ_0 \langle gekregen \circ_0 heeft\rangle^0)) \vdash s}\,PP2}{hij \circ_1 ((een \circ_1 ipadres) \circ_1 \langle toegewezen \circ_0 (gekregen \circ_0 heeft)\rangle^0) \vdash s}\,P2}{\vdots} \tag{69}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{ik \circ_1 ((((een \circ_1 woning) \circ_1 \langle toegewezen\rangle^p{}_0)\langle gekregen\rangle^p) \circ_0 \langle heb\rangle^0) \vdash s}\,PP1}{ik \circ_1 ((((een \circ_1 woning) \circ_1 \langle toegewezen\rangle^p{}_0)\langle gekregen\rangle^0) \circ_0 \langle heb\rangle^0) \vdash s}\,P4}{ik \circ_1 (((een \circ_1 woning) \circ_1 (\langle toegewezen\rangle^p \circ_0 \langle gekregen\rangle^0)) \circ_0 \langle heb\rangle^0) \vdash s}\,P4}{ik \circ_1 ((een \circ_1 woning) \circ_1 ((\langle toegewezen\rangle^p \circ_0 \langle gekregen\rangle^0) \circ_0 \langle heb\rangle^0)) \vdash s}\,PP3'}{ik \circ_1 ((een \circ_1 woning) \circ_1 (\langle toegewezen\rangle^p \circ_0 (\langle gekregen\rangle^0 \circ_0 \langle heb\rangle^0))) \vdash s}\,P3}{ik \circ_1 ((een \circ_1 woning) \circ_1 (\langle toegewezen\rangle^p \circ_0 \langle heb \circ_0 gekregen\rangle^0)) \vdash s}\,PP2}{ik \circ_1 ((een \circ_1 woning) \circ_1 \langle toegewezen \circ_0 (heb \circ_0 gekregen)\rangle^0) \vdash s}\,P2}{\vdots} \tag{70}$$

With this, we can accept all grammatical word orders of clusters with two participles. However, with these postulates, certain ungrammatical word orders of clusters with two participles are also accepted, see equation (71).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle toegewezen\rangle^p) \circ_0 \langle gekregen\rangle^p) \circ_0 \langle heb\rangle^0)) \vdash s}\,P4}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 (\langle toegewezen\rangle^p \circ_0 (\langle gekregen\rangle^p \circ_0 \langle heb\rangle^0)))) \vdash s}\,PP3'}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 (\langle toegewezen\rangle^p \circ_0 (\langle gekregen\rangle^p \circ_0 \langle heb\rangle^0)))) \vdash s}\,PP1}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 (\langle toegewezen\rangle^0 \circ_0 (\langle gekregen\rangle^p \circ_0 \langle heb\rangle^0)))) \vdash s}\,PP2}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 (\langle toegewezen\rangle^0 \circ_0 \langle gekregen \circ_0 heb\rangle^0))) \vdash s}\,P3}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle (gekregen \circ_0 heb) \circ_0 toegewezen\rangle^0)) \vdash s}\,P2}{\vdots} \tag{71}$$

Postulate PP1 is to blame for enabling the ungrammatical orders. Since PP1 can change $\Diamond_p$'s into $\Diamond_0$'s at any time during the derivation, P3 to move the main participle to the right of the other participle once the latter participle has lost its $\Diamond_p$. We attempt to solve this problem by modifying postulate PP1 so that it both moves the participle and

changes its $\diamondsuit_p$ into a $\diamondsuit_0$ at the same time, see PP1' (72).

$$PP1' : \diamondsuit_0(A \otimes_0 B) \to \diamondsuit_p B \otimes_0 \diamondsuit_0 A \tag{72}$$

Unfortunately, this doesn't solve our problem. Even with PP1' instead of PP1 the order in 70 can still be accepted, see the derivation in (73).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle toegewezen \rangle^p) \circ_0 \langle gekregen \rangle^p) \circ_0 \langle heb \rangle^0)) \vdash s}\, P4}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 (\langle toegewezen \rangle^p \circ_0 (\langle gekregen \rangle^p \circ_0 \langle heb \rangle^0)))) \vdash s}\, PP3'}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 (\langle toegewezen \rangle^p \circ_0 (\langle gekregen \rangle^p \circ_0 \langle heb \rangle^0)))) \vdash s}\, PP2}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 (\langle toegewezen \rangle^p \circ_0 \langle gekregen \circ_0 heb \rangle^0))) \vdash s}\, PP1'}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle (gekregen \circ_0 heb) \circ_0 toegewezen \rangle^0)) \vdash s}\, P2$$

$$\vdots \tag{73}$$

Even worse, with PP1' the word order in 23 will no longer be accepted, since the main participle cannot move past the finite verb because the other participle is in the way. Fortunately, this can be solved by introducing PP4 (74) that takes the $\diamondsuit_p$'s of the participles, and puts a $\diamondsuit_p$ around both participles, after which PP1' can move it (75).

$$PP4 : \diamondsuit_p(A \otimes_0 B) \to \diamondsuit_p A \otimes_0 \diamondsuit_p B \tag{74}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 (((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle (toegewezen \rangle^p \circ_0 \langle gekregen \rangle^p)) \circ_0 \langle heb \rangle^0)) \vdash s}\, PP4}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 (((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle toegewezen \circ_0 gekregen \rangle^p) \circ_0 \langle heb \rangle^0)) \vdash s}\, P4}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 (\langle toegewezen \circ_0 gekregen \rangle^p \circ_0 \langle heb \rangle^0))) \vdash s}\, PP1'}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle heb \circ_0 (toegewezen \circ_0 gekregen) \rangle^0)) \vdash s}\, P2$$

$$\vdots \tag{75}$$

Returning to our earlier problem, we introduced PP1 in order to enable participles to appear to the right of their finite or infinite selector, which, from a top-down perspective, would be the first word reordering in the derivation. Unintentionally, this postulate also allows verbs to be moved later in the derivation, which enables the main participle to appear to the right of its selector. We need to restrict this postulate so that it can only move the participle at the start of the derivation. A way of doing so is to make sure that before we start moving the participle up the tree structure, we change its mode so that PP1' will be unable to move this (even with this new mode, PP1 will still be able

to create an ungrammatical order (76), so we use PP1' in our final set of postulates).

$$\vdots$$
$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{ik \circ_1 ((sinds \circ_1 kort) \circ_1 (((een \circ_1 (ander \circ_1 ipadres)) \circ_1 (\langle toegewezen\rangle^p \circ_0 \langle gekregen\rangle^p)) \circ_0 \langle heb\rangle^0)) \vdash s}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 (((een \circ_1 (ander \circ_1 ipadres)) \circ_1 (\langle toegewezen\rangle^0 \circ_0 \langle gekregen\rangle^p)) \circ_0 \langle heb\rangle^0)) \vdash s}PP1}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle toegewezen\rangle^0) \circ_0 \langle gekregen\rangle^0) \circ_0 \langle heb\rangle^0)) \vdash s}PP1}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 (((een \circ_1 (ander \circ_1 ipadres)) \circ_1 (\langle toegewezen\rangle^0 \circ_0 \langle gekregen\rangle^0)) \circ_0 \langle heb\rangle^0)) \vdash s}P4}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 (((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle gekregen \circ_0 toegewezen\rangle^0) \circ_0 \langle heb\rangle^0)) \vdash s}P3}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 (\langle gekregen \circ_0 toegewezen\rangle^0 \circ_0 \langle heb\rangle^0))) \vdash s}P4}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle heb \circ_0 (gekregen \circ_0 toegewezen)\rangle^0)) \vdash s}P3}{\vdots}P2$$

(76)

We will call this new mode p1. With this mode, we can create our final set of postulates.

$$PP1 : \diamondsuit_0(A \otimes_0 B) \rightarrow \diamondsuit_p B \otimes_0 \diamondsuit_0 A \tag{77}$$

$$PP2 : \diamondsuit_{p1} A \rightarrow \diamondsuit_p A \tag{78}$$

$$PP3 : \diamondsuit_0(A \otimes_0 B) \rightarrow \diamondsuit_{p1} A \otimes_0 \diamondsuit_0 B \tag{79}$$

$$PP4 : \diamondsuit_{p1} A \otimes_0 (B \otimes_0 C) \rightarrow (\diamondsuit_{p1} A \otimes_1 B) \otimes_0 C \tag{80}$$

$$PP5 : \diamondsuit_p(A \otimes_0 B) \rightarrow \diamondsuit_p A \otimes_0 \diamondsuit_p B \tag{81}$$

$$PP6 : A \otimes_1 (B \otimes_0 \diamondsuit_p C) \rightarrow (A \otimes_1 B) \otimes_0 \diamondsuit_p C \tag{82}$$

Our new PP1 (77) is the same as PP1', but since the mode p now only appears at the top of the derivation, it will only reverse the order of the participle and its selector. The new PP2 (78) changes $\diamondsuit_p$ into $\diamondsuit_{p1}$, for the other postulates to work with. The new PP3 (79) and PP4 (80) are the same as our old PP2 and PP3' respectively, except they require a $\diamondsuit_{p1}$ rather than a $\diamondsuit_p$. The new PP5 (81) is the exact same as the old PP4. PP5 uses the mode p rather than mode p1 because it needs the p mode so PP1 can put both participles at the end of the cluster, see the derivation in (83), which is equivalent with the derivation in (75).

$$\vdots$$
$$\dfrac{\dfrac{\dfrac{\dfrac{ik \circ_1 ((sinds \circ_1 kort) \circ_1 (((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle(\langle toegewezen\rangle^p \circ_0 \langle gekregen\rangle^p)) \circ_0 \langle heb\rangle^0)) \vdash s}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 (((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle toegewezen \circ_0 gekregen\rangle^p) \circ_0 \langle heb\rangle^0)) \vdash s}PP5}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 (\langle toegewezen \circ_0 gekregen\rangle^p \circ_0 \langle heb\rangle^0))) \vdash s}P4}{ik \circ_1 ((sinds \circ_1 kort) \circ_1 ((een \circ_1 (ander \circ_1 ipadres)) \circ_1 \langle heb \circ_0 (toegewezen \circ_0 gekregen)\rangle^0)) \vdash s}PP1$$
$$\dfrac{}{\vdots}P2$$

(83)

Finally, PP6 (82) is a modification of P4, which allows the second verb in the cluster to be a participle rather than a finite or infinte verb, see equation (84) for an example.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{ik \circ_1 ((((een \circ_1 woning) \circ_1 \langle toegewezen)\rangle^p \circ_0 \langle gekregen\rangle^0) \circ_0 \langle heb\rangle^p) \vdash s}\, PP2}{ik \circ_1 ((((een \circ_1 woning) \circ_1 \langle toegewezen)\rangle^{p1} \circ_0 \langle gekregen\rangle^p) \circ_0 \langle heb\rangle^0) \vdash s}\, PP6}{ik \circ_1 (((een \circ_1 woning) \circ_1 (\langle toegewezen\rangle^{p1} \circ_0 \langle gekregen\rangle^p)) \circ_0 \langle heb\rangle^0) \vdash s}\, P4}{ik \circ_1 ((een \circ_1 woning) \circ_1 ((\langle toegewezen\rangle^{p1} \circ_0 \langle gekregen\rangle^p) \circ_0 \langle heb\rangle^0)) \vdash s}\, PP4}{ik \circ_1 ((een \circ_1 woning) \circ_1 (\langle toegewezen\rangle^{p1} \circ_0 (\langle gekregen\rangle^p \circ_0 \langle heb\rangle^0))) \vdash s}\, PP1}{ik \circ_1 ((een \circ_1 woning) \circ_1 (\langle toegewezen\rangle^{p1} \circ_0 \langle heb \circ_0 gekregen\rangle^0)) \vdash s}\, PP3}{ik \circ_1 ((een \circ_1 woning) \circ_1 \langle toegewezen \circ_0 (heb \circ_0 gekregen)\rangle^0) \vdash s}\, P2}{\vdots} \tag{84}$$

With these postulates the grammar accepts clusters with one or two postulates that have a grammatical word order, but rejects those that have an ungrammatical word order.

4 Discussion

With these postulates we can create a fragment that correctly parses the word order variation of verb clusters in Dutch subordinate clauses consisting of finite verbs, infinite verbs and participles.[3] However, the fragment can still be improved. The current fragment will accept clusters with three or more participles. However, there is an upper boundary to the number of participles in a Dutch cluster. Clusters with two participle arise either because there are both a perfect and a passive auxiliary in the same sentence, or because there is an auxiliary and a fossilized expression in the same cluster. Clusters with three participles would be theoretically possible if the cluster contains a perfect auxiliary, a passive auxiliary and a fossilized expression, but clusters with more than three participles are impossible because there are no other ways to create a participle and there cannot be two past auxiliaries, passive auxiliaries or fossilized expressions in the same cluster. The fragment will however accept these ungrammatical clusters because it does not have an upper boundary on the number of participles. Another improvement that can be made to the fragment is having it deal with cluster creepers. Cluster creepers are non-verbal elements that interrupt the verb cluster. Since the fragment is only made to be able to move verbs, it is unable to get these cluster creepers in their cannonical position. Outside the scope of this research, there are a number of further research possibilities. For example, extending the fragment in such a way that it can analyse te-infinitives, which behave differently from infinitives and participles. Another possibility for further research is extending the fragment so it can analyse non-subordinate clauses, for example declarative or interrogative clauses, containing verb clusters with participles as well. A final further research possibility is to create a fragment for other Germanic

[3] See Appendix A for the final version of the fragment

languages with verb clusters because other germanic languages have different rules for the order of verbs in verb clusters.

4.1 Conclusion

The research question for this study was 'How can the MTLG fragment by Michael Moortgat (1999) be extended so that it will be able to analyze verb clusters with past participles in Dutch subordinate clauses?' In this article, we have answered the question as follows: 'The MTLG fragment by Michael Moortgat (1999) can be extended with the six postulates presented in this article so that it is able to analyze verb clusters with past participles in Dutch subordinate clauses.' The research goal has therefore been achieved.

5 References

Augustinus, L. (2015). *Complement Raising and Cluster Formation in Dutch. A Treebank-supported Investigation.*

Bresnan, J., Kaplan, R. M., Peters, S., Zaenen, A. (1982). Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 13(4), 613-635. Broekhuis, H., & Corver, N. (2015). Syntax of Dutch: Verb and Verb Phrases. Volume 2.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on information theory, 2*(3), 113-124.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and control, 2*(2), 137-167.

Evers, A. (1975). *The transformational cycle in Dutch and German* (No. 64). Indiana University Linguistics Club.

Haeseryn, W. J. M., Romijn, K., Geerts, G., De Rooij, J., Van den Toorn, M. C. (1997). Algemene Nederlandse Spraakkunst [2 banden].

Huybregts, R. (1976). Overlapping dependencies in Dutch. *Utrecht Working Papers in Linguistics, 1*, 24-65.

Huybregts, R. (1984). The weak inadequacy of context-free phrase structure grammars. *Van periferie naar kern*, 81-99.

Moortgat, M. (1999). Meaningful patterns. *JFAK: Essays dedicated to Johan van Benthem on the occasion of his 50th birthday. Institute for Logic, Language, and Computation, University of Amsterdam. Available on CD-ROM at http://turing. wins. uva. nl.*

Moortgat, M. (2010). Typelogical grammar.

Muskens, R. (2007). Separating syntax and combinatorics in categorial grammar. *Research on Language & Computation, 5*(3), 267-285.

Shieber, S. M. (1985). Evidence against the context-freeness of natural language. *The Formal complexity of natural language, 33*, 320-332.

Stabler, E. (1997). Derivational minimalism. *Logical Aspects of Computational Linguistics*Steedman, M. (1985). Dependency and coördination in the grammar of Dutch and English. *Language*, 523-568., 68-95.

A The MTLG fragment

```
% -*- Mode: Prolog -*-
% ============================================================
% cluster_final.pl
% ============================================================
% !grail 3.1.1

% = Dutch verb clusters with past and passive participles.

:- abolish(lazy_unpack/1).
:- abolish(lazy_dr/1).
:- abolish(lazy_dl/1).
:- abolish(transparent_dia/1).
:- abolish(transparent/1).
:- abolish(continuous_dia/1).
:- abolish(continuous/1).
:- abolish(external_dia/1).
:- abolish(external/1).
:- abolish(postulate/3).
:- abolish(postulate1/3).
:- abolish(macro/2).
:- abolish(lex/3).
:- abolish(example/2).

:- dynamic lazy_unpack/1,lazy_dr/1,lazy_dl/1.
:- dynamic transparent_dia/1,transparent/1.
:- dynamic continuous_dia/1,continuous/1.
:- dynamic external_dia/1,external/1.
:- dynamic postulate/3,postulate1/3.
:- dynamic macro/2,lex/3,example/2.

% ============================================================
% Postulates
% ============================================================

% = structural postulates

postulate(p(1,A,zip(e,B)), zip(e,p(1,A,B)), 'P1').
postulate(zip(0,A), zip(e,A), 'P2').
postulate(p(0,zip(0,A),zip(0,B)), zip(0,p(0,B,A)), 'P3').
postulate(p(0,p(1,A,B),zip(0,C)), p(1,A,p(0,B,zip(0,C))), 'P4').
```

```
postulate(p(0,zip(p,A),zip(0,B)), zip(0,p(0,B,A)), 'PP1').
postulate(zip(p,A), zip(p1,A), 'PP2').
postulate(p(0,zip(p1,A),zip(0,B)), zip(0,p(0,A,B)), 'PP3').
postulate(p(0,p(0,zip(p1,A),B),C), p(0,zip(p1,A),p(0,B,C)), 'PP4').
postulate(p(0,zip(p,A),zip(p,B)), zip(p,p(0,A,B)), 'PP5').
postulate(p(0,p(1,A,B),zip(p,C)), p(1,A,p(0,B,zip(p,C))), 'PP6').

% = lazy evaluation

lazy_unpack(2).
lazy_unpack(i).

% = transparency

transparent(0).
transparent(2).
transparent(x).

transparent_dia(2).
transparent_dia(5).
transparent_dia(e).
transparent_dia(i).

% = continuity

continuous(2).
continuous(x).

continuous_dia(0).
continuous_dia(2).
continuous_dia(5).
continuous_dia(e).
continuous_dia(e1).
continuous_dia(i).
continuous_dia(w).

% = non internal modes

external(0).
external(1).
external(2).
external(x).
external(p).
```

```
external_dia(0).
external_dia(2).
external_dia(5).
external_dia(e).
external_dia(e1).
external_dia(i).
external_dia(w).

% ============================================================
% Macros
% ============================================================

% = macro(Form,Replacement)

macro(test, dl(1,np,dl(0,inf,inf))).
macro(bang(A,B), dia(A,box(A,B))).
macro(iv, dl(1,np,s)).
macro(tv, dl(1,np,iv)).
macro(dv, dl(1,pp,dl(1,np,iv))).
macro(rel, dl(1,n,n)).
macro(relprosub, dr(1,rel,dl(1,np,sub))).
macro(relpro, dr(1,rel,dl(1,bang(w,np),sub))).
macro(rpro, dr(1,pp,dr(1,pp,np))).
macro(qprom_r, dr(1,qm,box(i,dl(1,bang(w,rpro),s)))).
macro(qpros_r, dr(1,qs,dl(1,bang(w,rpro),box(e,s)))).
macro(relpro_r, dr(1,rel,dl(1,bang(w,rpro),sub))).
macro(qpro_main, dr(1,qm,box(i,dl(1,bang(w,np),s)))).
macro(qpro_sub, dr(1,xcs,dl(1,bang(w,np),box(e,s)))).
macro(dec, box(2,s)).
macro(sub, box(e,s)).
macro(xcs, dl(x,dl(1,bang(w,cs),s),s)).
macro(tepro, bang(5,te)).
macro(xte, dl(xt,dl(1,dia(w,box(5,te)),s),s)).
macro(prep, dr(1,pp,np)).
macro(ivmod, dr(1,iv,iv)).


% ============================================================
% Lexicon
% ============================================================

% = lex(Pros,Formula,Sem)
```

```
lex(het, dr(1,np,n), lambda(A,quant(iota,B,appl(A,B)))).
lex(de, dr(1,np,n), lambda(A,quant(iota,B,appl(A,B)))).
lex(een, dr(1,np,n), lambda(A,quant(iota,B,appl(A,B)))).
lex(die, dr(1,np,n), that).
lex(deze, dr(1,np,n), this).
lex(beide, dr(1,np,n), both).
lex(op, prep, on).
lex(voor, dr(1,dl(1,n,n),np), for).
lex(vanuit, dr(1,dl(1,np,np),np), from).
lex(in, dr(1,dr(1,iv,iv),np), in).
lex(op, dr(1,dr(1,iv,iv),np), on).
lex(bij, dr(1,dr(1,iv,iv),np), at).
lex(sinds, dr(1,dr(1,iv,iv),adv), since).
lex(daarom, dr(1,iv,iv), therefore).
lex(vanavond, dr(1,iv,iv), tonight).
lex(bent, box(0,dl(0,pastpart,iv)), are).
lex(is, box(0,dl(0,pastpart,iv)), is).
lex(is, box(0,dl(0,dl(1,np,pastpart),iv)), is).
lex(zijn, box(0,dl(0,pastpart,inf)), be).
lex(zijn, box(0,dl(0,dl(1,np,pastpart),inf)), be).
lex(geweest, box(p,dl(1,np,pastpart)), been).
lex(heeft, box(0,dl(0,pastpart,iv)), has).
lex(heeft, box(0,dl(p,pastpart,iv)), has).
lex(heb, box(0,dl(0,pastpart,iv)), have).
lex(hebben, box(0,dl(0,pastpart,inf)), have).
lex(worden, box(0,dl(0,pastpart,inf)), be).
lex(worden, box(0,dl(0,dl(1,np,pastpart),inf)), be).
lex(moeten, box(0,dl(0,inf,iv)), must).
lex(moeten, box(0,dl(0,inf,inf)), must).
lex(moeten, box(0,dl(0,inf,pastpart)), must).
lex(kunnen, box(0,dl(0,inf,inf)), can).
lex(kunnen, box(0,dl(0,inf,pastpart)), can).
lex(zullen, box(0,dl(0,inf,iv)), will).
lex(zouden, box(0,dl(0,inf,iv)), should).
lex(zou, box(0,dl(0,inf,iv)), should).
lex(omgekomen, box(p,pastpart), died).
lex(gekregen, box(p,dl(0,dl(1,pp,pastpart),pastpart)), gotten).
lex(toegewezen, box(p,dl(1,np,dl(1,pp,pastpart))), assigned).
lex(toegestuurd, box(p,dl(1,np,dl(1,pp,pastpart))), sent).
lex(aangeboden, box(p,dl(1,np,dl(1,pp,pastpart))), offered).
lex(verloren, box(p,dl(1,np,pastpart)), lost).
lex(vergeten, box(p,dl(1,np,pastpart)), forgotten).
```

```
lex(behandeld, box(p,dl(1,np,pastpart)), treated).
lex(uitgevoerd, box(p,dl(1,np,pastpart)), carriedout).
lex(bedekt, box(p,dl(1,np,pastpart)), covered).
lex(gegaan, box(p,dl(1,ap,pastpart)), gone).
lex(kort, adv, shortly).
lex(uw, dr(1,np,n), your).
lex(che, np, che).
lex(vandersteen, np, vandersteen).
lex(kaproen, np, kaproen).
lex(morel, np, morel).
lex(ckv, np, ckv).
lex(kinderen, np, children).
lex(lijsten, np, lists).
lex(contracten, np, contracts).
lex(kapot, ap, broken).
lex(noordelijke, dr(1,n,n), northern).
lex(ander, dr(1,n,n), other).
lex(verkiesbare, dr(1,n,n), eligible).
lex(vlaams, dr(1,n,n), flemish).
lex(reisdocument, n, lambda(A,appl(travel,appl(document,A)))).
lex(medicijnen, n, medicines).
lex(diensten, n, services).
lex(schouders, n, shoulders).
lex(reparaties, n, repairs).
lex(dag, n, day).
lex(woning, n, residence).
lex(ipadres, n, lambda(A,appl(ip,appl(adress,A)))).
lex(afschrift, n, transcript).
lex(haven, n, port).
lex(plaats, n, spot).
lex(buitenland, n, abroad).
lex(kartel, n, cartel).
lex(mei, dr(1,dl(1,num,np),num), may).
lex(ik, dr(1,s,iv), i).
lex(je, dr(1,s,iv), you).
lex(u, dr(1,s,iv), you).
lex(hij, dr(1,s,iv), he).
lex(zij, dr(1,s,iv), she).
lex(ze, dr(1,s,iv), she).
lex(ie, dr(1,s,iv), it).
lex(16, num, 16).
lex(2015, num, 2015).
lex(als, dr(1,als,sub), if).
```

```
lex(of, dr(1,of,sub), whether).
lex(dat, dr(1,dat,sub), that).
lex(indien, dr(1,indien,sub), incase).

% ============================================================
% Examples
% ============================================================

% = example(String,Formula)

example(" Als u uw reisdocument bent verloren.", als).
example(" Of u de medicijnen voor die dag vergeten bent.", of).
example(" Dat kinderen behandeld moeten worden.", dat).
example(" Dat de diensten zullen uitgevoerd zijn.", dat).
example(" Dat Che zou zijn omgekomen.", dat).
example(" Dat beide schouders bedekt zouden moeten zijn.", dat).
example(" Dat ie daarom kapot zou gegaan kunnen zijn.", dat).
example(" Dat Vandersteen Kaproen zou kunnen geweest zijn.", dat).
example(" Dat de reparaties in een noordelijke haven zouden moeten worden uitgevoerd.", d
example(" Dat hij een ipadres toegewezen gekregen heeft.", dat).
example(" Dat ik sinds kort een ander ipadres heb toegewezen gekregen.", dat).
example(" Of ik een woning heb toegewezen gekregen.", of).
example(" Dat je lijsten toegestuurd gekregen zou hebben.", dat).
example(" Dat ik op 16 mei 2015 een afschrift toegestuurd zou gekregen hebben.", dat).
example(" Dat zij deze woning toegewezen zou hebben gekregen.", dat).
example(" Als Morel bij het Vlaams kartel een verkiesbare plaats zou aangeboden hebben ge
example(" Indien ze vanavond contracten vanuit het buitenland zou aangeboden hebben gekre
example(" Als ik CKV zou aangeboden hebben gekregen.", als).
```

B Modified grail code for running the fragment

```
#grail0

#! /bin/bash

# Run grail kernel for one sentence
# $1 --> fragment (with path)
# $2 --> sentence
# $3 --> result type

# Adds the location of pdflatex to path
# PATH=$PATH:/usr/texbin/
if [ "$1" ] && [ "$2" ] && [ "$3" ]
```

```
then
if [ -e "$1" ]
then


        echo ""
        name=$(basename -s .pl $1)
        cd sources
        ./texcleanup.sh

  # echo prolog call to grail saved state
  echo "reset_statistics(off),load_fragment($name), tokenize(\"$2\",List), tex(List,$3).'

  # convert proof tex file to pdf
  if [ -f proofs1.tex ]; then
    pdflatex proofs1.tex > /dev/null

  # OS X specific, open pdf file
#  ps aux | grep Preview.app | grep -v grep; if [ $? = 0 ]; then open proofs1.pdf; fi
#    kill -9 Preview 2> /dev/null
    cygstart proofs1.pdf
  fi
  echo ""

else
    echo "..file $1 not found (enter PATH/FILENAME)"
fi
else
    echo "..Usage: grail [fragment (with path)] [sentence] [type]"
fi


#testall0

#! /bin/bash

# Run grail kernel for all example sentences of a fragment
# Only for grail simplified notation
# $1 --> fragment (with path)

if [ "$1" ]
then
if [ -e "$1" ]
then
```

```
  # grep all fragment subparts
  # add a space between /< (solves prolog tokenize error)
  # add inputcat.pl to handle simplefied notation
  #grep := $1 > fragments/tmp_macro
  #grep \# $1 > fragments/tmp_postulates
  #grep :: $1 | sed 's/\/</\/ </g' > fragments/tmp_lex
  #grep '===>' $1 > fragments/tmp_examples

  #cat sources/inputcat.pl fragments/tmp_macro fragments/tmp_postulates fragments/tmp_lex

  cp $1 fragments/tmp_testfrag.pl

  cd sources
  ./texcleanup.sh

  # echo prolog call to grail saved state
  # perform_all_tests tries to parse all example sentences
  echo "reset_statistics(off),perform_all_tests(tmp_testfrag)." | ./fragtest

  # remove temp files
  rm ../fragments/tmp_*

else
    echo "..file $1 not found"
fi
else
    echo "..Usage: testall [fragment (with path)]"
fi
```