Encoding Phases using Commutativity and Non-commutativity in a Logical Framework
Maxime Amblard

To cite this version:

Maxime Amblard. Encoding Phases using Commutativity and Non-commutativity in a Logical Framework. 6th International Conference on Logical Aspect of Computational Linguistic - LACL 2011, Jun 2011, Montpellier, France. pp.1–16, 10.1007/978-3-642-22221-4_1. hal-00601621

HAL Id: hal-00601621
https://hal.archives-ouvertes.fr/hal-00601621
Submitted on 25 Aug 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Encoding Phases using Commutativity and Non-commutativity in a Logical Framework

Maxime Amblard

Abstract. This article presents an extension of Minimalist Categorial Grammars (MCG) to encode Chomsky’s phases. These grammars are based on Partially Commutative Logic (PCL) and encode properties of Minimalist Grammars (MG) of Stabler [22]. The first implementation of MCG were using both non-commutative properties (to respect the linear word order in an utterance) and commutative ones (to model features of different constituents). Here, we propose to augment Chomsky’s phases with the non-commutative tensor product of the logic. Then we can give account of the PIC [7] just with logical properties of the framework instead of defining a specific rule.

Keywords: type theory, syntax, linguistic modeling, generative theory, phase, Partially Commutative Logic
In this paper, we propose a reconsideration of the properties of commutativity and non-commutativity in MCG to account the concept of phase introduced by Chomsky. Due to space consideration, we will limit ourselves to the problems of parsing, leaving aside the semantic aspects. However, it should be noted that the syntax-semantics interface of MCG contains all the necessary material for its integration.

We first discuss the need for the two different relations in MCG. Then, we define Minimalist Categorial Grammars (MCG). Based on these definitions, the third section presents and encodes phases in MCG, and shows the implementation of the PIC using only logical properties.

1 Commutativity vs Non-Commutativity in Standard MCG and Phases

To link logic and Generative Theory, MCG’s derivations are proofs of a restriction of Partially Commutative Logic (PCL), [20], seen as syntactic representations in generative theory. This logic is an extension of Lambek calculus containing simultaneously commutative and non-commutative connectives (ie introduction and elimination of implication and tensor). To handle the different relations between hypotheses in the same framework, an entropy rule (restriction of order) is added. Moreover,[3] shows a weak normalization of this calculus, to produce regular analysis in MCG.

All definitions of these grammars are given in [1]1, with a composition of rules (note that according to these definitions, normalisation is strong). Moreover, this restriction does not use introduction of hypothesis. They appear in the derivation only from specific lexical entries: in the hypotheses of a given category and that category as a formula. The lexicon will contain the entry $\phi \vdash \phi$ with $\phi$ as a given category.

The concrete part of the proof is achieved by a merge whose heart is the elimination of / or \ (rules that are found in different versions from categorial grammars , [12], [24] [16]) plus the entropy rule. This is one point where commutativity and non-commutativity play a crucial role. In particular, for the word order, it is clear that the relation is non-commutativity: being on the right or the left of a given word could not just be the same. Non-commutativity is also needed here, because of the second rule in MCG.

The hypotheses are seen as special position markers in the sequence of hypotheses of the proof. They are considered as resources of prominent features related to a phrase. They are unloaded by using the move operation of the generative theory. A direct implication is that the sequence of hypotheses in the proof contains exactly the sequence of available resources for further derivation. In this case, this sequence is a collection of resources and could not be a strict list. Unless a canonical order on the sequence of applied rules is presupposed, the only way to express this is with non-commutativity. The Merge rule must contain a release of the order. Definitions of basic rules of MCG involve commutativity and non-commutativity. We will show how we could use these properties in another perspective to encode a linguistic concept, more precisely by noting that the changing category of the verb controls the process flow.

---

1 These definitions contain also a syntax / semantic interface. However, we leave this part out of this article due to space.
Chomsky’s theory introduces the notion of phase, which corresponds to the evolution of the verb. Thus, we assume that the lexical item associated with the verb carries only part of its achievement in the sentence. This is an important use in the syntax-semantic interface for reporting UTAH \(^2\). The needed hypothesis to introduce a DP is provided by verb lexical items. And this is the correspondence of verb’s resources with the DP category which allows the DP to be in the proof.

In the phase’s definition, Chomsky assumes that some instance of move (unification of features in our formalism) must be completed before the end of the phase. Once it is reached, the specific resources of the process are no longer accessible to the rest of the analysis. It defines an island in the analysis, called Phase Impenetrability Condition (PIC). Translating this definition into our formalism implies that a phase (or its representation) block access to part of the sequence of hypotheses of the proof. We assume that the interpretation in MCG is a non-commutative point in the sequence of hypotheses.

The direct implication is that the analysis of a sentence simultaneously uses relations to link noun and verb phrases, and non-commutative relations to construct the analysis (changing the category of the verb). The use of non-commutativity involves a strict order to control the verb’s role in the analysis. Formal properties used by items of different categories are disjointed and it allows to fully exploit their relations.

### 2 Minimalist Categorial Grammars

Minimalist Categorial grammars are based on a fragment of PCL to encode the MP of Chomsky. Then it uses an abstract calculus to produce both a string (sequence of words) and a semantic representation (formula). Word order and semantics are synchronized over the main calculus which uses a structured lexicon and specific rules. First, we briefly present rules of PCL, then we introduce labels which encode word order, then we define lexical items and finally introduce rules of MCG.

#### 2.1 Partially Commutative Logic (PCL)

The logic introduced in [8] and extended in [21] is a superimposition of the Lambek calculus (Intuitionistic Non-Commutative Multiplicative Linear Logic) and Intuitionistic Commutative Multiplicative Linear Logic. Connectives are:

- the Lambek calculus connectives: \(\odot, \setminus, /\) (non-commutatives)
- commutative multiplicative linear connectives: \(\otimes, \leftrightarrow\) (commutatives)

The use of these connectives is presented in figure 1.

In this logic, commutative and non-commutative relations could be used simultaneously. Then contexts are partially ordered multisets of formulae and in order to relax this order, we need an entropy rule noted \(\sqsubset\) which is defined as the replacement of \(;\) by \(,\). The restriction of elimination rules and the entropy rule is called Minimalist Logic (the logic used to define MCG). In the following, we note \(F\) the set of categories (\(F\) stands for features).

\(^2\) Uniform Theta Assignment Hypothesis or assignment of thematic roles.
Derivations of MCG are labelled proofs of the PCL. Before defining labelling, we define labels and operations on them. To do this, we use the set of phonological form \(Ph\) and a set \(V\) of variables such that: \(Ph \cap V = \emptyset\). We note \(T\) the union of \(Ph\) and \(V\). We define the set \(\Sigma\), called the labels set as the set of triplets of elements of \(T^*\). Every position in a triplet has a linguistic interpretation: they correspond to specifier/head/complement relations of minimalist trees. A label \(r\) will be considered as \(r = (r_s, r_h, r_c)\).

We introduce variables in the string triplets and a substitution operation. They are used to modify a position inside a triplet by a specific material. Intuitively, this is the counterpart in the phonological calculus of the product elimination.

A substitution is a partial function from \(V\) to \(T^*\). For \(\sigma\) a substitution, \(s\) a string of \(T^*\) and \(r\) a label, we note respectively \(s.\sigma\) and \(r.\sigma\) the string and the label obtained by the simultaneous substitution in \(s\) and \(r\) of the variables with the values associated by \(\sigma\) (variables for which \(\sigma\) is not defined remain the same).

If the domain of definition of a substitution \(\sigma\) is finite and equal to \(x_1, \ldots, x_n\) and \(\sigma(x_i) = t_i\), then \(\sigma\) is denoted by \([t_1/x_1, \ldots, t_n/x_n]\). Moreover, for a sequence \(s\) and a label \(r\), \(s.\sigma\) and \(r.\sigma\) are respectively denoted \(s[t_1/x_1, \ldots, t_n/x_n]\) and \(r[t_1/x_1, \ldots, t_n/x_n]\). Every injective substitution which takes values in \(V\) is called renaming. Two labels \(r_1\) and \(r_2\) (respectively two strings \(s_1\) and \(s_2\)) are equal modulo a renaming of variables if there exists a renaming \(\sigma\) such that \(r_1.\sigma = r_2\) (resp. \(s_1.\sigma = s_2\)).

Finally, we need another operation on string triplets which allows to combine them together; the string concatenation of \(T^*\) is noted \(\bullet\). Let \(\text{Concat}\) be the operation of concatenation on labels which concatenates the three components in the linear order: for \(r \in \Sigma\), \(\text{Concat}(r) = r_s \bullet r_h \bullet r_c\).

We then have defined a word order structure which encodes specifier/complement/head relations and two operations (substitution and concatenation). These two operations will be counterparts in the phonological calculus of \(\text{merge}\) and \(\text{move}\).
Labelled proofs. Before exhibiting the rules of MCG, we need to define the concept of labelling on a subset of rules of the Minimalist Logic ($\setminus_{e}$, $\setminus_{e}$, $\otimes_{e}$ and $\sqcap$).

For a given MCG $G$, let a $G$-background be $x : A$ with $x \in V$ and $A \in F$, or $\langle G_1; G_2 \rangle$ or else $\langle G_1, G_2 \rangle$ with $G_1$ and $G_2$ some $G$-backgrounds which are defined on two disjoint sets of variables. $G$-backgrounds are series-parallel orders on subsets of $V \times F$. They are naturally extended to the entropy rule, noted $\sqcap$. A $G$-sequent is a sequent of the form: $\Gamma \vdash_G (r_s, r_t, r_c) : B$ where $\Gamma$ is a $G$-background, $B \in F$ and $(r_s, r_t, r_c) \in \Sigma$.

A $G$-labelling is a derivation of a $G$-sequent obtained with the following rules:

\[
\begin{align*}
\langle s, A \rangle & \in \text{Lex} \quad \boxed{\text{Lex}} \\
\Gamma \vdash_G (\epsilon, s, \epsilon) : A & \quad \boxed{\text{axiom}} \\
\Gamma \vdash_G r_1 : A / B \quad \Delta \vdash_G r_2 : B & \quad \text{Var}(r_1) \cap \text{Var}(r_2) = \emptyset \quad \boxed{\setminus_{e}} \\
\langle \Gamma; \Delta \rangle \vdash_G (\text{Concat}(r_2)) : A & \\
\Delta \vdash_G r_2 : B & \quad \Gamma \vdash_G r_1 : B \setminus A \quad \text{Var}(r_1) \cap \text{Var}(r_2) = \emptyset \quad \boxed{\setminus_{e}} \\
\langle \Gamma; \Delta \rangle \vdash_G (\text{Concat}(r_2) \bullet r_{1s}, r_{1t}, r_{1c}) : A & \\
\Gamma \vdash_G r_1 : A \otimes B & \quad \Delta[x : A, y : B] \vdash_G r_2 : C \quad \text{Var}(r_1) \cap \text{Var}(r_2) = \emptyset \quad A \in P_2 \quad \boxed{\otimes_{e}} \\
\Delta[\Gamma] \vdash_G [\text{Concat}(r_1) / x, \epsilon / y] : C & \\
\Gamma \vdash_G r : A & \quad \Gamma' \sqcap \Gamma & \quad \boxed{\sqcap} \\
\Gamma' \vdash_G r : A & \\
\end{align*}
\]

Note that a $G$-labelling is a proof tree of the Minimalist Logic on which sequent hypotheses are decorated with variables and sequent conclusions are decorated with labels. Product elimination is applied with a substitution on labels and implication connectors with concatenation (a triplet is introduced in another one by concatenating its three components).

2.3 Lexicon

MCG encodes informations in the lexicon with types. They are defined over two sets, one of linguistic categories and the other of move features. Lexical items associate a label and a formula of PCL with respect to the following grammar:

\[
\begin{align*}
\text{L} & ::= (B) / P_1 \mid C \\
\text{B} & ::= P_1 \setminus (B) \mid P_2 \setminus (B) \mid C \mid D \\
\text{C} & ::= P_2 \otimes (C) \mid C_1 \\
\text{D} & ::= P_2 \otimes (D) \mid C_1 \\
\text{C}_1 & ::= P_1
\end{align*}
\]
where \( t \) is the starting non-terminal and \( P_1 \) and \( P_2 \) are atomic formulae belonging to set of features of the MCG (features which trigger \textit{merge} or \textit{move} rules).

Formulae of lexical items get started with a \( / \) as the first connective, and continue with a sequence of \( \backslash \). This corresponds to the sequence of selectors and licensors in MG lexical items. These are trigger rule features of MCGs. They give the concrete part of the derivation. A formula is ended with an atomic type, the category of the phrase, or a sequence of \( \odot \) (which contains at least a specific type, which is also the main category).

For example, the following formula could be the one of an MCG entry:

\[
(c_m \ldots \backslash c_1 \langle b_1 \odot \ldots \odot b_n \odot a \rangle) / d
\]

with \( a \in P_1, b_i \in P_2, c_j \in P \) and \( d \in P_1 \).

The morphism from MG lexicon to MCG ones is defined in [1].

### 2.4 Rules of MCG

In the same way as for MG, [22], rules of MCG are defined over two principles:

- combining two pieces of derivations: \textit{merge}
- redefining internal relations in a derivation: \textit{move}

As we have mentioned before, MCG is defined over a restriction of PCL: elimination of \( / \) and \( \backslash \), and \( \odot \). In the following, in order to distinguish relations in the sequence of hypotheses, a commutative relation will be marked with ‘;’ and a non-one with ‘;’.

- \textit{Merge} is the function which combines two pieces of proofs together and it needs an non-commutative relation to correctly encode relations among words. But in the same application, \textit{merge} will also combine hypothesis of different proofs. And from a linguistic point of view, relations between these hypothesis should be commutative because there is no reason to block access to them. Then, \textit{merge} combines an elimination of \( \backslash \) or \( / \) with the application of an entropy rule. For the same linguistic reasons as in MG, MCG use two different kinds of \textit{merge} depending on the lexical/non-lexical status of the trigger.

For the word order, \textit{merge} is simply the concatenation of the string of one phrase in the label of the other one (depending of right/left relation):

Lexical trigger:

\[
\begin{align*}
\vdash (r_s, r_h, r_c) & : A / B \quad \Delta \vdash s : B & \quad \text{[}/c]\text{]} \\
\Delta \vdash (r_s, r_h, r_c \bullet Concat(s)) : A & \quad \text{[entropy]} \\
\Delta \vdash (r_s, r_h, r_c) : A & \quad \text{[mq]} \\
\end{align*}
\]
A *merge* with a lexical item do not explicitly show the order between hypotheses. But here, the entropy rule is the replacement of a hypothesis by a

Non-lexical trigger:

\[
\Delta \vdash s : B \quad \Gamma \vdash (r_s, r_h, r_c) : B \setminus A \\
\Delta; \Gamma \vdash (\text{Concat}(s) \bullet r_s, r_h, r_c) : A \quad \text{[entropy]} \\
\Delta, \Gamma \vdash (\text{Concat}(s) \bullet r_s, r_h, r_c) : A \\
\implies \\
\Delta \vdash s : B \quad \Gamma \vdash (r_s, r_h, r_c) : B \setminus A \\
\Delta, \Gamma \vdash (\text{Concat}(s) \bullet r_s, r_h, r_c) : A \quad \text{[mg]}
\]

- The encoding of *Move* in MCG is structurally different from the one in MG. Here, it assumes that a phrase is included in the proof if and only if all its hypotheses are. Then, we do not really reinterpret the local tree as in MG, but we directly produce the final derivation tree. In this way, a *move* is the discharge of hypotheses by a `⊗`

For word order, the concatenation of the moved phrase is substituted in the position of the newest hypothesis:

\[
\Gamma \vdash r_1 : A \otimes B \quad \Delta[u : A, v : B] \vdash r_2 : C \\
\Delta[\Gamma] \vdash r_2[\text{Concat}(r_1)/u, v] : C \quad \text{[mv]}
\]

Finally, to give account of [23], the framework is enriched with rules which modifies the position of the string in the label. There are two kinds of rules:

- **head movement** where the head of the merged element is concatenated in the final head. This implies four rules: two to distinguish left and right concatenation and two over the lexical status of the trigger of *merge*.
- **Affix hopping** where the head of the trigger is concatenated with the head of the merged element. In the same way, there are four rules to distinguish left from right concatenations and lexical status of the trigger.

We use a simple way to encode the different possibilities of merging with `< and >`.

Pointing to the connective indicates head-movement and outside defines affix hopping.

The lexical status does not need to be represented. In the following of this paper, only head movement will be used, thus we give these four rules:

**Head-Movement:**

\[
\Gamma \vdash (r_{\text{spec}}, r_{\text{tete}}, r_{\text{comp}}) : A / < B \quad \Delta \vdash s : B \\
\Delta[\Gamma] \vdash r_{\text{spec}}, r_{\text{tete}}, r_{\text{comp}} \otimes \text{Concat}(s_{\text{tete}}) : A \quad \text{[mg]} \\
\Gamma, \Delta \vdash (r_{\text{spec}}, r_{\text{tete}}, r_{\text{comp}}) : A > / B \quad \Delta \vdash s : B \\
\Gamma, \Delta \vdash (r_{\text{spec}}, r_{\text{tete}}, r_{\text{comp}}) \bullet \text{Concat}(s_{\text{tete}}) : A \quad \text{[mg]} \\
\Delta \vdash s : B \quad \Gamma \vdash (r_{\text{spec}}, r_{\text{tete}}, r_{\text{comp}}) : B \setminus A \\
\Delta, \Gamma \vdash (\text{Concat}(s_{\text{tete}}) \bullet r_{\text{spec}}, r_{\text{tete}}, r_{\text{comp}}) : A \quad \text{[mg]} \\
\Delta \vdash s : B \quad \Gamma \vdash (r_{\text{spec}}, r_{\text{tete}}, r_{\text{comp}}) : B \setminus < A \\
\Delta, \Gamma \vdash (\text{Concat}(s_{\text{tete}}) \bullet r_{\text{spec}}, r_{\text{tete}}, r_{\text{comp}}) : A \quad \text{[mg]} 
\]
3 Phases

3.1 Encoding Phases in MCG

Following [7], Chomsky assumes that the analysis of a sentence is driven by the verb which goes by two specific states: the phases VP and cP. Note that neither tP nor the decomposition over simple verb form as it is used in usual MCG are phases (this is illustrated in figure 2). Moreover, Chomsky claims that syntactic islands are defined by phases. That is, the content of a phase must be moved to its left-hand side in order to let it accessible. This step of the phase is called a transfer.

![Figure 2. Phases in verb structure](image)

According to this structure, items on the right side of the phase can not be moved again. This syntactic island is called the Phase Impenetrability Condition (PIC). The definition of phases and the PIC are still under debate. Nevertheless, an interesting point for MCG is the simultaneous use of commutative and non-commutative properties of the framework to recognize them.

According to Chomsky, a phase is a node of the syntactic tree which triggers allowed moves. Because it is a node, this implies for MCG to combine two subproofs. It is not possible to simulate this with a single lexical item (which implies a terminal node). Also transferring is the realization of possible moves and the direct linking of hypotheses in a cyclic move.

Another argument in support of phases is in the semantic counterpart. This article does not present this second synchronized part of the calculus in MCG. Many different works claim that at particular step of the verb derivation some specific thematic roles could be assigned as in [10], [11] and [4]. [1] and [14] describe both arguments for this and the semantic tiers according to these assumptions. But, they do not include the idea of phases. Only remarks that simplification of derivation requires specific points in which continuation reductions must occur. We leave the presentation of consequences of phases on semantics to future works.

The analysis of a simple sentence derivation in standard MSG uses 4 items. In the following, the verb *read* will be used to illustrate this presentation for a simple active affirmative sentence.
1. The deep syntactic build of the verb: the verb and all its arguments (except the subject).
   \[ \vdash v \not< d \]
2. Mode: introduces the subject category and, at least, the accusative case.
   \[ \vdash k \not< d \not< V / v \]
   \[ \vdash k \not< t / v \]
3. Inflection: brings the inflection to the verb.
   \[ \vdash k \not< t / V \]
4. Comp: fully completes the analysis (a question mark, insert in relative clause, etc.)
   \[ \vdash c / t \]

In order to keep control on the string associate to the verb, the derivation systematically uses head movement except comp which ends the derivation. We claim that phases can be encoded in MCG with a non-commutative order. The lexical realization of a phase item explicitly contains hypotheses in a non-commutative order.

\[ \Delta_1, H_1; H_2 \Delta_2 \vdash A \]  

(1)

This order makes a strong boundary in the derivation of the phase. It blocks move of elements in complement position to specifier. These are the only items that are lexically built with hypothesis different from the original definition of MCG [15], [1] and [2].

Using the \( \odot_e \) rule of the PCL, the phase rule is defined as:

\[
\begin{array}{c}
\Delta_s, \Delta_h, \Delta_c \vdash (s_s, s_h, s_c) : X \odot Y \\
\Gamma_s, X; Y, \Gamma_c \vdash (r_s, r_h, r_c) : Z
\end{array}
\]

\[ [\text{phase}] \]

The phase rule is the combination of a discharge of hypothesis in non-commutative order and a transfer step. This transfer is the realization of all possible moves and parts of cyclic ones. This new rule assembles several individual rules of PCL proof. It may be difficult to follow the derivation step by step. In the following, phases are given with more details. Thus, we note \([\text{phase}_1] \) the substitution part of the phase (the use of \( \odot_e \) which combines two proofs) and \([\text{phase}_{\text{trans}}] \) moves which can be achieved after \([\text{phase}_1] \). The main condition to validate a phase is \( \Delta_c \) and \( \Gamma_c \) are empty after \([\text{phase}_{\text{trans}}] \). This correspond to a phrase in complement position of a phase does not stand accessible. We note MCGphase, MCG wit phases.

There is a direct consequence of this encoding of phase on the structure of the lexical item Hypotheses on its left-hand side of take the place of the complement of the head. Thus, all elements with which it should be combined must be in a specifier position. A more complex formula as in (1) will be built only with \( \not< \):

\[ \Delta_1, H_1; H_2 \Delta_2 \vdash (s_s, s_h, s_c) B_n \not< \ldots \not< B_0 \not< A \]  

(2)

To take into account of the definition of the phases theory and the proposed encoding in MCG, a simple sentence will be divided into two phases: one with the mode and another with comp. Their lexical items are modified in this way into:

- mode: \( \vdash k \not< d \not< V / v \Rightarrow V; v \vdash k \not< d \not< V \)
- comp: \( \vdash c / t \Rightarrow c; t \vdash c \)
Hypotheses on the left-hand side of formulae receive a straightforward interpretation. They correspond to the conversion of a proof of a $v$ to a proof of a $V$ (or from a $t$ to a $c$ in the second one). Thus, we need to update the two other formulae in order to combine them with the two previous ones:

- **verb:** $\vdash v / d \Rightarrow \vdash (V \circ v) / d$
- **inflection:** $\vdash k \backslash t / V \Rightarrow \vdash k \backslash (c \circ t) / V$

Note that the use of $\circ$ in a formula do not imply that this item is one of a *phase*. It means that they take an active part in the construction of the verb. Here, we have extended the main category of the item to the previous one, combined with the category of the following *phase* category with a $\circ$.

In fact, each head item drives a specific part of all elements included under their relation to the head. But, the unloading of hypotheses (the realization of the *phase*) occurs later. Here, the logical account of the framework updates the derivation in a way that there is no direct intuition with the syntactic tree. We present a simple example of interpretation of a *phase* in a proof into a tree: on one hand the starting part of the derivation and on the other the proof which results in the *phase*:

\[
\begin{align*}
\vdash (A \circ B) / C & \quad C \vdash C \\
\text{[mg]} & \quad A; B \vdash D \backslash E \\
\end{align*}
\]

Which correspond to the two following syntactic trees:

- $\vdash (A \circ B) / C$
- $C \vdash C$
- $D \vdash D$
- $A; B \vdash D \backslash E$
- $\epsilon$

The syntactic tree of the *phase* is not a simple leaf but a tree with an empty position in which the other derivation is substituted:

- $\vdash (A \circ B) / C$
- $C \vdash C$
- $D \vdash D$
- $A; B \vdash D \backslash E$
- $\epsilon$

And it is clear that all information derived in the second tree before the *phase* combination must be updated after. Finally, for the same reasons as for *merge*, Head Movement and Affix Hopping are needed through the *phase*. The same connectives are used with the same effects. One instance is:

\[
\begin{align*}
\Delta_s, \Delta_h, \Delta_c \vdash (s_s, s_h, s_c) : X \circ_c Y \\
\Gamma_s, X; Y, \Gamma_c \vdash (r_s, r_h, r_c) : Z \\
\end{align*}
\]

\[
\begin{align*}
\Gamma_s \Delta_s, \Delta_h \vdash (r_s \bullet s_s, s_h \bullet r_h, s_c \bullet r_c) : Z \\
\text{[phase]}
\end{align*}
\]
where the head of the triplet is the concatenation of the two heads. The following section presents an application of this rules in a full analysis of a simple sentence.

### 3.2 Derivation of a simple sentence

To illustrate derivation with *phases*, this section presents the complete derivation of a simple sentence. However, extending the analysis to more complex syntactic structures has resulted in defining the lexical entries corresponding with the same principles. We present the analysis of the following example:

(1) The children read a book.

This simple sentence uses the affirmative form, the verb will take the four previous steps. In order to build noun phrase with MCG with generative theory, we combine a determiner $\vdash (k \otimes d) / n$ with a noun $\vdash n$. It produces a constituent of category $d$ (for *determinal phrase*) which lacks the assignment of syntactic case ($k$). Then, the following lexicon is used:

| articles         | $\vdash (\epsilon, \text{the}, \epsilon) : k \otimes d / n$
|                 | $\vdash (\epsilon, a, \epsilon) : k \otimes d / n$
| noun             | $\vdash (\epsilon, \text{children}, \epsilon) : n$
|                 | $\vdash (\epsilon, \text{book}, \epsilon) : n$
| verb (mode)      | $\vdash (\epsilon, \text{read}, \epsilon) : (V \otimes v)_{<} / d$
| (inflection) (comp) | $V; v \vdash (\epsilon, \epsilon, \epsilon) : k \setminus d \setminus V$
|                 | $\vdash (\epsilon, -, \epsilon) : k \setminus (\epsilon \odot t) / _{<} V$
|                 | $c; t \vdash (\epsilon, \epsilon, \epsilon) : c$

The derivation is a proof, so each main element (the head of each phrases) drives its subproof. The consequence is that the proof is built in several parts adjusted against each other by the discharge of hypotheses. This property makes the presentation more complex, but the search for proofs (*i.e.* the parsing) could be done in parallel. In the following, each verb step is presented. Note that the normalization of PCL from [3] ensures that we could combine each step one by one as presented here.

**Step 1** In the first verb step of the derivation, the first position of the verb is saturated by a hypothesis of category $d$. It corresponds to the position of the main component that occupies the object position in the sentence. The object is not directly inserted in the derivation because all its features are not yet marked by hypotheses: the position of $k$ (mark of the syntactic case assignment\(^3\)) is missed. This is a departure from MG, in which all phrases are directly inserted in the derivation. Here, we only mark positions for insertion.

\[
\begin{align*}
\vdash (\epsilon, \text{read}, \epsilon) : (V \otimes v)_{<} / d & \quad d \vdash (\epsilon, u, \epsilon) : d \\
\vdash (\epsilon, \text{read}, u) : (V \otimes v) & \quad [mg]
\end{align*}
\]

\(^3\) in the semantic part of the calculus, this position is also used for the thematic role assignment.
This is the end of the first verb step. The result must be inserted in another proof which contains hypothesis $V$ and $v$. On one hand, the interpretation of the non-commutative relation between $V$ and $v$ in the type could be the saturation of all arguments of an element of type $v$ to produce a $V$.

**Step 2** These hypotheses are in the proof driven by the second entry of the verb: mode which ends the first phase. First, we need to saturate all positions of mode with lexical hypothesis, and then we could:

1. combine the result with the first verb step,
2. introduce the object of the sentence with a move in the transfer part of the phase.

The lexical item of mode is merged with a hypothesis $k$ and next a $d$:

$$
\begin{align*}
  & k \vdash (\epsilon, v, \epsilon) : k \\
  & v \vdash (\epsilon, \epsilon, \epsilon) : d \setminus V
\end{align*}
$$

$$
\begin{align*}
  & d \vdash (\epsilon, w, \epsilon) : d \\
  & k, V \vdash (v, \epsilon, \epsilon) : d \setminus V
\end{align*}
$$

At this point of the derivation, this result is combined with the first verb step with a [phase] with head movement. Note that the string of the head of the discharged is concatenated to the string of the head position in order to keep all structural information over the verb accessible to the full derivation. The substitution part of the phase produces:

$$
\begin{align*}
  & d \vdash (\epsilon, read, u) : (V \odot v)_< \\
  & d, k, V \vdash (w v, \epsilon, \epsilon) : V
\end{align*}
$$

Before ending the phase, we perform a move with all positions of the utterance’s object (they are now in the proof). Then it could be introduced in the derivation with a transfer. In parallel, the determiner phrase is built with the two lexical entries ”a” and “book” by a merge:

$$
\begin{align*}
  & \vdash (\epsilon, a, \epsilon) : (k \otimes d) / n \\
  & \vdash (\epsilon, book, \epsilon) : n
\end{align*}
$$

Thereby, it is discharged in the main proof. The choice of the $d$ with which to carry out the unloading is not left by chance. The derivation must be continued with the one which empties the previous verb step with a move. This is exactly the interpretation of transfer part in [7].

$$
\begin{align*}
  & \vdash (\epsilon, a, book) : k \otimes d \\
  & d, k, d \vdash (w v, read, u) : V
\end{align*}
$$

This move substitute in the newest variable as define in [1]. It is the full realization of the constituent. In this phase, non-commutativity and commutativity are both used. Non-commutativity in order to keep the structure of the verb and commutativity to unload hypotheses of the nominal phrase. This underlies the assumption that the order of the features of the noun could not be presupposed. This is reinforced by the analysis of questions where that object constituent undergoes one more move and then must be explicitly transfered from right to left part of the phase. Now, it is the end of the second verb step. The derivation continues by preparing the third one.
Step 3  This part of the derivation must be combined with the next lexical entry of the verb, the inflection. In this part of the verb step, it was merged with the previous result and next with a \( k \) hypothesis - the position of the subject case:

\[
\begin{align*}
\Gamma &\vdash (e, -, e) : k \setminus (c \odot t) / \prec V \\
\Gamma &\vdash (w \ a \ book, \ read, e) : V \\
k, d &\vdash (z, read, w \ a \ book) : (c \odot t)
\end{align*}
\]

This allows to discharge hypothesis about the subject constituent which it also build:

\[
\begin{align*}
\Gamma &\vdash (\epsilon, \ the, \ e) : (k \otimes d) / n \\
\Gamma &\vdash (\epsilon, \ children, \ e) : n \\
\Gamma &\vdash (\epsilon, \ the, \ children) : k \otimes d
\end{align*}
\]

And unloaded:

\[
\begin{align*}
\Gamma &\vdash (\epsilon, \ the, \ children) : k \otimes d \\
k, d &\vdash (z, read, w \ a \ book) : (c \odot t)
\end{align*}
\]

Step 4  This example stands for a very simple sentence, then the last verb step corresponds only to the combination of the current bypass with the lexical entry \( \text{comp} \) by a phase with nothing in the transfer part:

\[
\begin{align*}
\Gamma &\vdash (\epsilon) : c \\
\Gamma &\vdash (\epsilon, \ e) : c \\
\Gamma &\vdash \epsilon ; t \vdash (\epsilon, \ e, \ e) : c \\
\Gamma &\vdash (\epsilon, \ e, \ the \ children \ read \ a \ book) : c
\end{align*}
\]

This ends the last phase and thus the derivation. The proof matches the string \textit{The children read a book}. An important distinction with the previous versions of these grammars is in the use of lexical item without phonological part. Here, only the lexical items used in the phase process are necessary, but the structure of items for phases imposes a strict order in their pooling.

3.3 Question

In the previous example, the transfer part of the two phases is not really efficient. The first one introduced the object of the utterance and the second only ended the derivation. For questions, the \( \text{comp} \) item is more complex because it introduces the last feature of the object. This time, its lexical item is \( \epsilon ; t \vdash (\epsilon, \ e, \ e) : wh \setminus c \). And it is only afterward that a hypothesis \( wh \) is introduced that the object could be introduced in the derivation. But it means that in the previous phase, the constituent mark must be transferred from the left to the right part of the first phase. This is done with a cyclic move: the introduction of a new hypothesis \( k \otimes d \vdash k \otimes d \), which explicitly connect the two hypotheses.

In the lexicon, only \( \text{comp} \) is modified and an item for which is added:

\[
\begin{align*}
\Gamma &\vdash \epsilon, \ e, \ e : (wh \otimes (k \otimes d))
\end{align*}
\]

The derivation before the phase is still the same.
1. First step procedure:

\[
\begin{align*}
\vdash (\epsilon, \text{read}, \epsilon) : (V \odot v) / d & \quad d \vdash (\epsilon, \text{read}, u) : (V \odot v) \\
\vdash (\epsilon, \text{u}, \epsilon) : d & \quad k \vdash (\epsilon, \text{v}, \epsilon) : d \backslash V \\
\vdash (\epsilon, \text{read}, u) : (V \odot v) & \\
\end{align*}
\]

2. Saturation of positions of mode:

\[
\begin{align*}
\vdash (\epsilon, \text{w}, \epsilon) : d & \quad k \vdash (\epsilon, \text{v}, \epsilon) : d \backslash V \\
\vdash (\epsilon, \text{v}, \epsilon) : d & \quad k \vdash (\epsilon, \text{v}, \epsilon) : d \backslash V \\
\vdash (\epsilon, \text{u}, \epsilon) : d & \quad k \vdash (\epsilon, \text{v}, \epsilon) : d \backslash V \\
\vdash (\epsilon, \text{v}, \epsilon) : d & \quad k \vdash (\epsilon, \text{v}, \epsilon) : d \backslash V \\
\end{align*}
\]

3. Construction of the object constituent:

\[
\begin{align*}
\vdash (\epsilon, \text{which}, \epsilon) : (\text{wh} \otimes (k \otimes d)) / n & \quad \vdash (\epsilon, \text{book}, \epsilon) : n \\
\vdash (\epsilon, \text{which}, \text{book}) : (\text{wh} \otimes (k \otimes d)) \\
\end{align*}
\]

We get all the necessary material to process the phase. Its first part combines:

\[
\begin{align*}
\vdash (\epsilon, \text{read}, u) : (V \odot v) & \quad d, k, V \vdash (w v, \epsilon, \epsilon) : V \\
\vdash (\epsilon, \text{read}, u) : (V \odot v) & \quad d, k, V \vdash (w v, \epsilon, \epsilon) : V \\
\end{align*}
\]

In the treatment of this utterance, the transfer part is not able to directly discharge the two hypotheses of the determiner phrase. A cyclic move is used in order to store the access to this element. At the same time, the phase move it on its left part:

\[
\begin{align*}
\vdash (\epsilon, \text{which}, \epsilon) : (\text{wh} \otimes (k \otimes d)) / n & \quad \vdash (\epsilon, \text{book}, \epsilon) : n \\
\vdash (\epsilon, \text{which}, \text{book}) : (\text{wh} \otimes (k \otimes d)) \\
\end{align*}
\]

\[
\begin{align*}
\vdash (\epsilon, \text{which}, \text{book}) : (\text{wh} \otimes (k \otimes d)) & \\
\vdash (\epsilon, \text{which}, \text{book}) : (\text{wh} \otimes (k \otimes d)) \\
\end{align*}
\]

Finally, before the last phase, the derivation introduces a wh hypothesis which will allow the move of object after the first part of the phase realization. The move of the transfer part is:

\[
\begin{align*}
\vdash (\epsilon, \text{which}, \text{book}) : (\text{wh} \otimes (k \otimes d)) & \quad \vdash (\epsilon, \text{book}, \epsilon, \text{the children read}) : c \\
\vdash (\epsilon, \text{which}, \text{book}) : (\text{wh} \otimes (k \otimes d)) & \quad \vdash (\epsilon, \text{book}, \epsilon, \text{the children read}) : c \\
\end{align*}
\]

In this example, only the transfer in the first phase, which accounts for the cyclic move of the constituent allows to complete the derivation.

### 3.4 Blocked derivation with PIC

A very important point in the definition of the phase rule is the fact that the complementizer part of hypothesis must be removed. This property encodes the Phase Impenetrability Condition. The previous one does not contain such problem because the transfer part of the first phase achieves all moves which empty complementizer hypotheses.

A simple example extracted from the previous one is the case where the $k$ hypothesis in the second step of the verb is not included. Thus the derivation must failed because one hypothesis is away. The lexical entry corresponding to mode is:
\[ V; v \vdash (\epsilon, \epsilon, \epsilon) : d \setminus V \]

which produces a conclusion of a proof of type \( V \) with only \( d \) in the left hand side:

\[ d, V; v \vdash (w\epsilon, \epsilon) : V \]

The result of the first part of the phase is:

\[ d \vdash (\epsilon, \text{read}, u) : (V \odot v) \]

\[ d, V; v \vdash (w v, \epsilon, \epsilon) : V \]

\[ d, d \vdash (w v, \text{read}, u) : V \quad [\text{phase1}] \]

And the transfer part does not contribute to this step. The part of the \( \Gamma_c \) of the phase rule is not removed. The structure of the proof which blocks the derivation is the case where the constituent is in complementizer position of the head.

We would remark that derivations with phases immediately block the process unlike traditional MCG or MG which perform the full derivation before concluding that a specific feature stand at the end (and reject the derivation).

Even if this example is quite simple, it shows that the encoding of PIC directly uses properties of the MCG. Unlike the other constraints, we do not need to propose new rules. That insure to keep the same generative power for MCG\textsubscript{phase}. The derivation strictly controls the structure and check internal relations.

4 Conclusion

The main aim of this paper is to introduce the concept of phase from minimalism into type logical grammars, simulating the generative theory of Chomsky which has been an open question since [7]. It involves the introduction of a new rule into the system, and highlights commutative and non-commutative relationships between elements of the parsing process. Moreover, this addition is not ad hoc as it allows full use of the properties of PCL underlying the formalism. This new rule is the composition of a substitution of hypotheses in commutative relations, followed by a transfer that is either the realization of a move became possible, or a cyclic move. This proposal also involves a new linguistic interpretation of cyclic move.

A full description of the system would require additional details [1]. However, we emphasize the role played by phases at the syntactic level to define islands where the encoding of PIC is simply the use of logical properties of the framework. Furthermore, we claim that the use of phases at semantic level corresponds to the introduction of thematic role predicates (variables related to the reification of formulas and substitution of variables). They also mark points in the semantic tiers where the context must be reduced. It plays a crucial role at the semantic level by marking reduction point for continuations.

The introduction of these phases confirms the use of a logical system simultaneously handling relations commutative and non-commutative at plays in linguistic analysis. Distributing the properties on each component used in this analysis can produce fine performances A remaining issue for this description is the formalization of the phase as reduction point at the semantic level that would reduce ambiguities of scope of quantifiers. In addition, the study of equivalence between the MCG with phases and MG remains an open question.
Acknowledgments

The author would like to express his gratitude to reviewers for their precise remarks, Corinna Anderson, Sai Qian and Sandrine Ribeau for their readings.

References