Extending Tree-adjoining grammars and Minimalist Grammars with unbounded scrambling: an overview of the problem area

Alexander Perekrestenko

Universitat Rovira i Virgili
Departamento de Filologías Románicas
Grupo de Investigación en Lingüística Matemática

Pl. Imperial Tarraco 1; 43005 Tarragona

alexander.perekrestenko@estudiants.urv.cat

Abstract

In this paper we give an overview of restricted formalisms used as a basis for formal syntactic theories and discuss the problem of extending these formalisms with an adequate representation of nonlocal scrambling, the phenomenon of structural discontinuity leading to the intermingling of the fragments belonging to different syntactic constituents.

Keywords: syntax, restricted formalisms, Tree-adjoining Grammars, Minimalist Grammars, scrambling

Resumen

En este artículo presentamos unos formalismos restringidos utilizados como base para distintas teorías formales de sintaxis y tratamos el problema de la extensión de estos formalismos para la representación correcta del fenómeno sintáctico llamado scrambling no local. Este fenómeno de discontinuidad estructural hace que se entremezclen fragmentos de diferentes constituyentes sintácticos.

Palabras clave: sintaxis, formalismos restringidos, gramáticas de adjunción de árboles, gramáticas minimalistas, scrambling

Resum

En aquest article presentem uns formalismes restringits usats com a base per a diferents teories sintàctiques formals i tractem el problema de l'extensió d'aquests formalismes per a la representació correcta del fenomen sintàctic conegut com a scrambling no local. Aquest fenomen de discontinuïtat estructural fa que es barregin fragments pertanyents a diferents constituents sintàctics.

Paraules clau: sintaxi, formalismes restringits, gramàtiques d'adjunció d'arbres, gramàtiques minimalistes, scrambling

Table of contents

1. Introduction

Syntax as a theory describes how words of a language combine to form bigger units, sentences, with which we communicate. Human language differs from other communication systems, such as the "language" of animals or artificial sign systems, in that it allows compositionality of meaning that is possible due to its powerful structure composition device which we call syntactic component. In the 20th century, a number of syntactic theories have emerged that viewed syntax from different perspectives and used different methodology to describe the object of their study. The application of exact mathematical models in syntactic research began on a large scale in the 1950s with Noam Chomsky's Transformational Grammar. Since then a number of theoretical syntactic frameworks with different degree of formalization/formalizability have been proposed, of which Lexical-Functional Grammar (LFG), Generalized Phrase Structure Grammar (GPSG), Head-driven Phrase Structure Grammar (HPSG), Meaning-Text Theory (MTT), Tree-adjoining Grammars (TAG), Government and Binding Theory (GB), and the Minimalist Program have had the biggest impact on syntactic research.

In the description of syntax there are two basic approaches possible with respect to what aspects of the structure the backbone component of the theory is intended to capture. Sentences can be described in terms of their *constituent structure* or in terms of the *dependencies* existing between the elements of the sentence, whatever these elements might be. In the first approach, every syntactic structure is considered as being composed of smaller units which are composed of other even smaller units and so on, until we arrive at the lexical elements. This way of specifying syntax is used in the so-called *phrase structure grammars*. A typical representative of this class of syntactic frameworks is the Transformational Grammar in its early versions.[1] The latter approach normally does not imply the existence of intermediate composite structures and operates directly on lexical elements, describing syntactic structures in terms of the relationships existing between the words they contain. Frameworks of this kind are usually referred to as *dependency grammars*. A framework that is predominantly based on this approach is the syntactic component of the Meaning-Text Theory (Mel'čuk 1974, 1988). Each of these approaches captures only certain aspects of syntax whereas both the constituent structure and the structure of dependencies are necessary in order to obtain a complete picture of how a language is organized on the syntactic level. Most of the theories of syntax take into account both constituent and dependency structure. In this paper we will limit ourselves to the representation of the constituent structure and to the corresponding formalizations.

The mentioned syntactic frameworks with the exception of GB and, to a certain extent, Minimalist Program and MTT are based on strict mathematical concepts. In this respect, a special status can be attributed to Tree-adjoining Grammars which are actually not a syntactic theory as such, but rather a purely mathematical formalism that has proved to be well-suited for the description of natural language syntax. They were originally inspired in the GB and its predecessor theories, but came later to be used as an independent syntactic framework (Frank 2002).

Another promising formalism that potentially can be used for practical syntactic description are Minimalist Grammars (MG) proposed in (Stabler 1997) as a formal tool

[1] The result of its evolution known as Government and Binding Theory (Chomsky 1995) is no longer so one-sidedly constituent-structure-oriented.

for modeling the fundamental structure-building operations of the Minimalist Program, an approach adopted within the Chomskyan branch of syntactic theory (Chomsky 1995, 2001). The comparatively lesser popularity of MGs compared to TAGs can be explained with a certain bias of MGs towards the Chomskyan concept of syntax while TAGs appear to be more theory-independent. Nevertheless, MGs can be used as a convenient formal device for the representation of syntax for practical purposes like grammar engineering and parsing.

2. Formalisms

The following classes of formalisms usually come into consideration as a base for syntactic theories:

Right-linear (regular) grammars. These grammars can only be used for so-called *shallow parsing*. They are not powerful enough to describe natural languages even in the weak sense, i.e., in terms of their terminal sequences. This idea can be informally illustrated by the following example of embedded infinitival clauses in German:

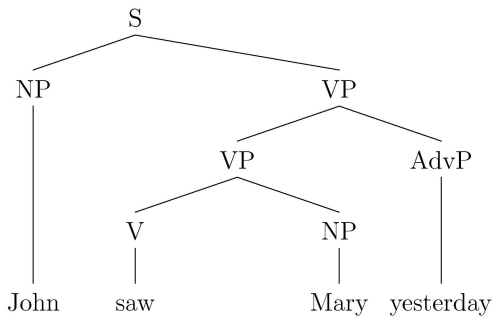
(er is bereit) die Kinder das Haus anstreichen zu lassen
(he is ready) $[\text{the children}]_{NP}$ $[\text{the house}]_{NP}$ $[\text{to paint}]_{V}$ $[\text{to let}]_{V}$
'(he is ready) to let the children paint the house'

There is potentially no limit on the number of embedded infinitival clauses:

(er is bereit) Hans die Kinder das Haus anstreichen zu lassen zu versprechen
(he is ready) $[\text{Hans}]_{NP}$ $[\text{the children}]_{NP}$ $[\text{the house}]_{NP}$ $[\text{paint}]_{V}$ $[\text{to let}]_{V}$ $[\text{to promise}]_{V}$
'(he is ready) to promise Hans to let the children paint the house'

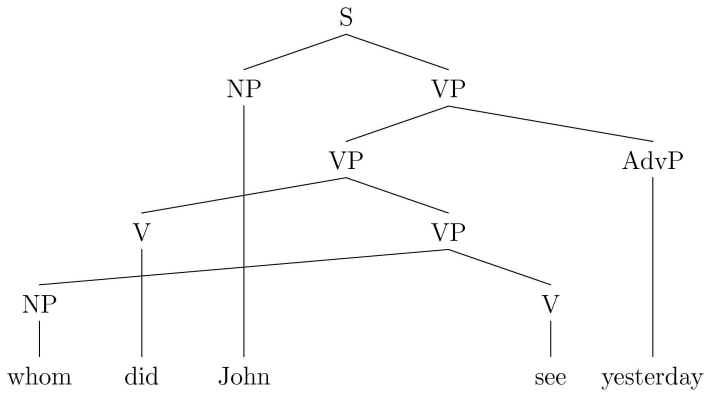
In this class of sentences, we have the non-regular language $\{ NP^i V^i \mid i \in \mathbf{N} \}$.[2] Since regular grammars do not support arbitrary recursion, they cannot be used as a formal basis of a syntactic theory.

Context-free grammars. Although context free grammars (CFG) can describe a big part of the natural language sentences in the weak sense, i.e., as strings of words, they fail to assign appropriate structural descriptions to sentences containing discontinuous constituents. Using CFG rules we can describe the structure of the sentence consisting of continuous constituents only, like for example *John saw Mary yesterday*:



[2] A more rigorous argument would involve an intersection with a regular languages and a morphism.

But it is impossible to produce an adequate descriptions for sentence with discontinuous constituents, like *Whom did John see yesterday?* if we only have a CFG-based descriptional device:



This is the reason why pure CFG-based formalisms without extensions cannot be used in constituent based syntactic frameworks.[3] On the other hand, CFGs have several attractive properties: they are easy to use in parsing, their properties are well studied, and they make it possible to describe the vast majority of syntactic structures of natural languages at least in the weak sense, that means that they only need a "slight" extension in order to become suited for syntactic description.

Mildly context-sensitive grammars. Formalisms of this class have their motivation in linguistics as there are several phenomena in natural languages requiring for their generalized description formal devices which are more powerful than CFGs, even in the weak sense. There has been much discussion of these phenomena in the past, so we will not go into the details of these cases here.

A class **G** of grammars is said to be mildly context-sensitive if it satisfies the following conditions:

1. **G** includes all context-free languages, and there are grammars G_1, G_2, and G_3 in **G** generating the languages $\{ ww \mid w \in \Sigma^* \}$, $\{ a^m b^n c^m d^n \mid a,b \in \Sigma^*, m,n \in \mathbf{N} \}$ and $\{ a^n b^n c^n \mid a,b,c \in \Sigma^*, n \in \mathbf{N} \}$.
2. For every grammar $G \in \mathbf{G}$, the language L(G) has a constant growth property.
3. For every grammar $G \in \mathbf{G}$ and a word $w \in L(G)$, the membership problem ¿$w \in G(L)$? is decidable in deterministic polynomial time with respect to the length of w.

The best studied and most widely used mildly context-sensitive syntactic formalisms are local *Tree-adjoining Grammars* (TAGs) and *Minimalist Grammars* (MGs).

Mildly context-sensitive grammars were proposed as a *constrained* formalism for the description of natural language syntax. The idea was to create a formal system that

[3] Though, it should be noted that arguments about syntactic structures are always based on the stipulations of the syntactic theory in question. In this particular case it means that within the commonly accepted view of what a syntactic constituent is, no acceptable description of this sentence can be given using CFG rules only. This in principle does not exclude a possibility that with other syntax-theoretical assumptions this example could very well be rendered by a CFG-based framework.

would only be powerful enough to describe the syntactic structures of natural languages and nothing else and that would therefore serve as an exact mathematical model of the syntactic component of the language.

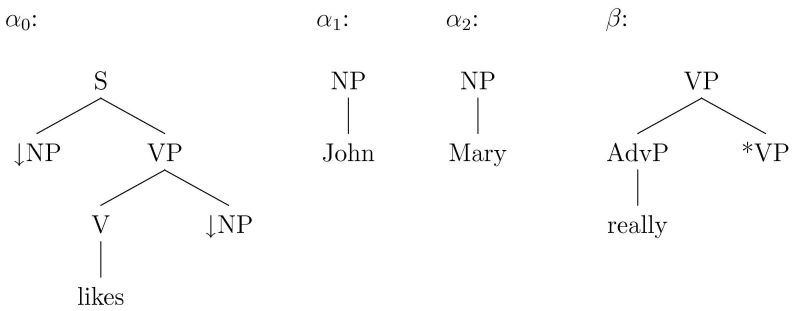
Computationally unrestricted formalisms. Unification-based syntactic frameworks with unrestricted structure sharing strictly speaking do not belong to the class of restricted grammars since they are based on unification formalisms which are Turing-equivalent. The problem of the computational universality of the formalism itself is solved with the design of grammars that do not exploit the full power of the formalism. In this case, computational universality is certainly not motivated linguistically, but rather the way the formalism is constructed appears convenient for its use within the syntactic theory which in principle "does not care" about the computational properties of the formalism it is based upon as long as the formalism provides it with all the means necessary for the intuitively appealing description of syntactic structures and allows a polynomial-time processing for linguistically adequate descriptions. HPSG can be mentioned as an example of a computationally restricted grammar framework based on a Turing-equivalent formalism (a unification grammar).

3. Tree-adjoining grammars

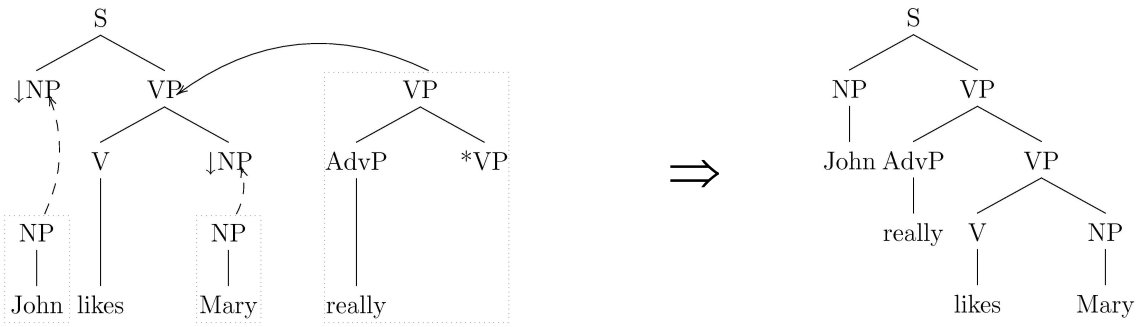
Tree-adjoining grammars (TAG) are the best known and the most widely used mildly context-sensitive syntactic formalism. In linguistics, normally lexicalized TAGs are used. Elementary structures of a TAG are trees. In a lexicalized TAG, each tree has a lexical anchor associated with it. Each such tree represents a lexical entry. Lexicalized TAGs do not use rules as the way syntactic structures are constructed is coded in the lexical entries.

The definitions of the so-called pure tree-adjoining grammars and tree-adjoining grammars with adjunction constraints based on (Joshi 1997) are given in the Appendix 1.

Below we give the example of a TAG generating the sentence *John really likes Mary*. The grammar contains the trees α_0, α_1, α_2, and β:



The derivation proceeds as illustrated in the following picture where the trees α_1 and α_2 are attached into the tree α_0 by means of the *substitution* and the tree β is inserted into α_0 by *adjunction*:



A number of extensions to TAGs have been proposed in order to cope with syntactic phenomena unprocessable with single-component TAGs. Some of these extension are Multicomponent TAGs (MCTAG) introduced in (Joshi 1987; Weir 1988) that can be tree-local (tlMCTAG), set-local (slMCTAG) and non-local (nlMCTAG), vector TAGs with dominance links and integrity constraints (VTAG-Δ) (Rambow 1994), Multicomponent TAGs with shared nodes (SN-MCTAG) (Kallmeyer 2005) and Multicomponent TAGs with tree tuples (TT-MCTAG) (Lichte 2007).

4. Scrambling

Originally, *scrambling* was the name for the kind of the argument permutation observed in the so-called *middlefield* (Mittelfeld) in German. However, this phenomenon is also attested in many other languages, e.g., in Hindi, Japanese, Korean, Russian and Turkish. Here we will mostly illustrate scrambling on German examples, but the results can probably without much change be extended to other languages. In German, the NP and PP arguments (i.e., subject(s) and objects), roughly speaking, can be freely permutated in the space between:

- a finite verb on the left and the leftmost non-finite verb, a verbal particle or the nonverb part of an idiomatic expression on the right, or
- a complementizer on the left and the leftmost verb or the nonverb part of an idiomatic expression on the right.

The computationally most problematic case of this phenomenon is the so-called *unbounded nonlocal scrambling* consisting in the permutation of the arguments belonging to different verbal heads. In this kind of scrambling, a change in the linear order of the constituents is the result of the displacement of some of them from their infinitival clauses into the matrix clause. Since the depth of the infinitival clause embedding is potentially unlimited, we can have any number of arbitrarily ordered arguments "jumping up" from embedded infinitival clauses to the matrix clause as shown in the following examples based on (Rambow 1994). All the sentences of this example mean 'that no-one has tried to promise the customer to repair the refrigerator':

...dass niemand [[*dem Kunden*] [[*den Kühlschrank*] *zu reparieren*] *zu versprechen*] *versucht hat*
...that no-one [[the customer] [[the refrigerator] to repair] to promise] tried has

...dass niemand [*den Kühlschrank*]$_i$ [[*dem Kunden*] [t_i *zu reparieren*] *zu versprechen*] *versucht hat*
...that no-one [the refrigerator]$_i$ [[the customer] [t_i to repair] to promise] tried has

...dass [*den Kühlschrank*]$_i$ *niemand* [[*dem Kunden*] [t_i *zu reparieren*] *zu versprechen*] *versucht hat*
...that [the refrigerator]$_i$ no-one [[the customer] [t_i to repair] to promise] tried has

...dass [*dem Kunden*]$_j$ *niemand* [t_j [[*den Kühlschrank*] *zu reparieren*] *zu versprechen*] *versucht hat*
...that [the customer]$_j$ no-one [t_j [[the refrigerator] to repair] to promise] tried has

...dass [*den Kühlschrank*]$_i$ [*dem Kunden*]$_j$ *niemand* [t_j [t_i *zu reparieren*] *zu versprechen*] *versucht hat*
...that [the refrigerator]$_i$ [the customer]$_j$ no-one [t_j [t_i to repair] to promise] tried has

...dass [*dem Kunden*]$_j$ [*den Kühlschrank*]$_i$ *niemand* [t_j [t_i *zu reparieren*] *zu versprechen*] *versucht hat*
that [the customer]$_j$ [the refrigerator]$_i$ no-one [t_j [t_i to repair] to promise] tried has.

In German, long-distance scrambling can only proceed from non-tensed clauses. Phrases in which the boundary of a tensed clause is crossed are perceived as incorrect:

**Peter hat* [*den Kühlschrank*]$_i$ *versprochen, dass er* t_i *reparieren wird*
Peter has [the refrigerator]$_i$ promised, that he t_i repair will
Intended reading: 'Peter has promised that he will repair the refrigerator'

Arguments cannot scramble out of non-complementized tensed clauses either so that the scrambling renders a strongly deviant structure in the following case as well:

**Peter hat* [*den Kühlschrank*]$_i$ *gesagt, er wird* t_i *reparieren*
Peter has [the refrigerator]$_i$ said, he will t_i repair
Intended reading: 'Peter has said he will repair the refrigerator'

Thus a syntactic formalism extended with scrambling must also provide for the possibility to restrict scrambling in the way indicated above.

As was shown in (Becker 1992), unbounded scrambling cannot be adequately described by local MCTAGs. Vector TAGs with dominance links and integrity constraints (VTAG-Δ) seem to be the only formalism capable of modeling scrambling with barriers in a way that also permits polynomial-time recognition, provided some restrictions are imposed on the derivation which are satisfied if the formalism is lexicalized. It is not known what class of languages a VTAG-Δ without this restrictions is able to generate, but it can be conjectured that the recognition problem for them will most probably be NP-hard. Generally, the situation with the formalization of scrambling is such that all known TAG-based formalisms able to provide a general (or close-to-general) account of scrambling are either

- NP-hard – see e.g. (Champollion 2007) for the NP-completeness proof for a restricted version of nlMCTAGs and (Søgaard 2007) for the NP-hardness results on free-order TAGs, unrestricted SN-MCTAGs and TT-MCTAGs), or

- they can only account for scrambling from imbedded clauses of a depth limited by a constant – the result on SN-MCTAG-k in (Kallmeyer 2005) provides an example of this restriction, or
- (apparently) require additional restrictions on the derivation, external to the formalism itself (Rambow 1994).

At this moment, no TAG-based formalism is known that would account for unbounded nonlocal scrambling in the generalized way and that at the same time would be provably free from the above-mentioned limitations.

5. Minimalist Grammars

Minimalist Grammars (MGs) were proposed in (Stabler 1997) as a formal tool for modeling some fundamental structure-building operations of the Minimalist Program (Chomsky 1995, 2001). Unrestricted Minimalist Grammars introduced in (Stabler 1997) belong to the class of mildly context-sensitive grammar formalisms and are weakly equivalent to slMCTAGs. This follows from the results in (Michaelis 1998; 2001), and (Harkema 2001).

The "standard" MG uses two structure-building operations – *merge* and *move*. The *merge* operator combines trees to produce new trees in a way structurally similar to the substitution in TAGs. The *move* operator displaces a subtree whose head is *licensed for movement* into another position within the tree. Following the principles of the Minimalist approach, movement is assumed to be triggered by the necessity of one category to "check off" its features against some other category. The failure of a category to check off its features as well as the failure of a category to act as a checker for another one makes the derivation crash.

The generative power of MGs largely depends on the presence or absence of the so-called *locality constraints* (LCs). Two best investigated LCs in terms of their effect on the weak generative capacity of MGs are the *shortest-move constraint* (SMC) and the *specifier island constraint* (SPIC). The SMC prohibits competitive displacement of constituents, while the SPIC bars displacements from within constituents in specifier positions. Since movement goes to specifier positions, SPIC also bars displacements from within constituents that have moved. The so-called unrestricted MGs proposed in (Stabler 1997) only use SMC. In (Michaelis 2005) it was proved that adding SPIC to the MSC-restricted MG makes it less powerful. On the other hand, using only SPIC without SMC makes the formalism Turing-equivalent (Kobele 2005).

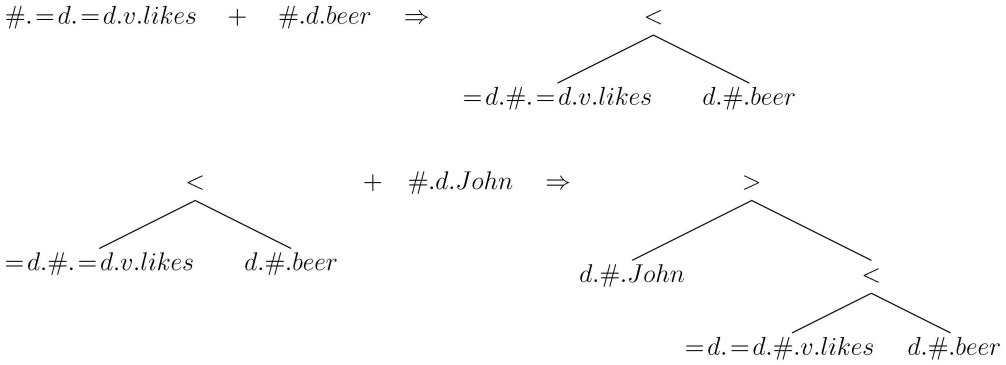
In order to model some specific syntactic operations not incorporated into the original version of MG, the formalism was extended with the operation of *scrambling* and *(cyclic) adjunction* in (Frey 2002) and *countercyclic adjunction* in (Michaelis 2003). However, the scrambling operator introduced in (Frey 2002) was restricted by SMC which reduced it to an operation similar to non-obligatory movement making the generalized description of this syntactic phenomenon impossible. From the formal point of view, the way SMC is introduced for movement does not make much sense for scrambling as an SMC-restricted scrambling can be simulated by movement. In order to get a linguistically meaningful definition of scrambling, the formalism must also implement barriers.

A mathematically strict definition of MG with unbounded nonlocal scrambling and nondiscriminating barriers is given in Appendix 2. It follows the definition in (Perekrestenko 2008) which is based on the definitions proposed in (Gärtner 2007) extended with barriers and the corresponding modification of the scrambling operator.

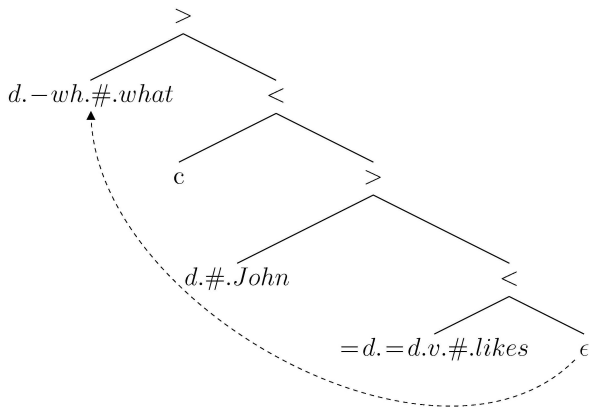
From the linguistic perspective, the principal difference between MGs and TAGs consists in the way they treat discontinuous constituents. TAGs use an extended domain of locality so that each tree in a TAG (or a tree set in an MCTAG)[4] corresponds to a single constituent. Eventual adjunction into such a tree "splits" it making in this way the constituent it represents discontinuous. In MGs, the discontinuity of a constituent is a result of the displacement of a part of the constituent into some other position in the tree. In this manner, MGs do not employ the notion of locality used in TAGs.

Below we will illustrate the derivation in MG with two examples.

Example 1. Lexicon: *#.=d.=d.v.likes; #.d.John; #.d.beer*
Derivation of the phrase *John likes beer* using the *merge* operator:



Example 2. Lexicon: *#.=d.=d.v.likes; #.d.John; #.d.-wh.what; #.=v.+wh.c*
Fragment of the derivation of the subordinate clause *…what John likes* using the *move* operator displacing the DP *what* from the object position of the verb *likes*.



In (Perekrestenko 2007; 2008), MGs were extended with scrambling and barriers. Scrambling as defined there operates basically like a movement, but it is optional and not limited by SMC. Moreover, it is restricted by barriers – subtrees out of which other

[4] In MCTAGs, tree sets are sometimes seen as trees with relaxed dominance relation.

constituents cannot scramble. The definition presented in the first paper uses category sensitive barriers blocking scrambling of specific categories while the extension to MGs defined in latter one employs nondiscriminating barriers blocking scrambling of any category.

As was shown in (Perekrestenko 2008), extending an MG with unrestricted scrambling and barriers makes the fixed recognition problem for the resulting formalism NP-hard even if only nondiscriminating barriers are used.

In (Gärtner 2007), scrambling was defined as extraposition to the right whereas the scrambling operator introduced in (Perekrestenko 2008) displaces constituents to the left, like movement. We conjecture that changing the direction of scrambling will not alter the validity of the result presented in the latter paper.

Another interesting question raised in (Gärtner 2007) is the effects of the so-called Adjunct Island Condition (AIC) disallowing constituent displacements from within adjuncts in the MG extended with countercyclic adjunction. As was indicated in (Michaelis 2003), countercyclic adjunction can circumvent the SMC, but adding the AIC to an SMC-restricted MG with countercyclic adjunction makes this "trick" impossible (Gärtner 2007).

As can be seen comparing the MG formalisms in (Gärtner 2007) and in (Perekrestenko 2008), an MG with unrestricted scrambling defined in the latter paper can be modeled by an SMC-restricted MG with countercyclic adjunction and without AIC, as presented in the first paper, provided no barriers are used. It is not clear in how far barriers affect the complexity of the recognition problem for the MG extended with unrestricted scrambling. It might happen that the universal recognition problem for an MG with scrambling and without barriers will be NP-hard while the fixed recognition problem might be in **P**.

6. Conclusions

As it turns out, MGs despite being derivationally different from TAGs do not contain any inherent "remedy" against the consequences of introducing unbounded scrambling with barriers into an originally polynomially processable formalism. It is not yet clear if it is possible to modify MGs with scrambling and barriers in a way that would allow polynomial-time processing and a generalized description of scrambling with barriers at the same time. This question remains to be investigated.

7. References

Becker, Tilman, Aravind Joshi, and Owen Rambow (1991). "Long Distance Scrambling and Tree Adjoining Grammars". In *Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics*, Berlin, Germany.

Becker, Tilman, Owen Rambow, and Michael Niv. 1992. "The derivational generative power, or, scrambling is beyond LCFRS". Technical report, University of Pennsylvania.

Champollion, Lucas (2007). Lexicalized non-local MCTAG with dominance links is NP-complete. In *Proceedings of Mathematics of Language 10*, UCLA. CSLI On-Line Publications. To appear.

Chomsky, Noam (1995). *The Minimalist Program*. The MIT Press.

Chomsky, Noam. 2001. "Derivation by phase". In M. Kenstowicz, ed. (2001). *Ken Hale: A Life in Language*. Cambridge, MA: MIT Press.

Frank, Robert (2002). *Phrase Structure Composition and Syntactic Dependencies*. The MIT Press.

Frey, Werner, and Hans-Martin Gärtner (2002). "On the treatment of scrambling and adjunction in minimalist grammars". In Jäger, G., P. Monachesi, G. Penn, and S. Wintner, eds. (2002). *Proceedings of Formal Grammar 2002*, Trento, 41-52.

Gärtner, Hans-Martin, and Michaelis, Jens (2007). "Some remarks on locality conditions and minimalist grammars". In Sauerland, U., and H.-M. Gärtner, eds.(2007). *Interfaces + Recursion = Language? Chomsky's Minimalism and the View from Syntax and Semantics*. Mouton de Gruyter, Berlin, 161-195.

Harkema, Henk (2001). "A characterization of Minimalist languages". In de Groote, Ph., G. Morrill and Chr. Retoré, eds. (2001). *Proceedings of the 4th International Conference on Logical Aspects of Computational Linguistics*, LACL'2001, France.

Joshi, Aravind K. (1987). "An introduction to tree adjoining grammars". In A. Manaster-Ramer, ed. (1987). *Mathematics of Language*. John Benjamins, Amsterdam, 87-114.

Joshi, Aravind K., and Yves Schabes (1997). "Tree-Adjoining Grammars". In Rozenberg, G., and A. Salomaa, eds. (1997). *Handbook of Formal Languages*, Vol. 3, Berlin, New York: Springer, 69-124.

Kallmeyer, Laura (2005). "Tree-Local Multicomponent Tree Adjoining Grammars with Shared Nodes". *Computational Linguistics* 32(2): 187-225.

Kobele, Gregory, and Jens Michaelis (2005). "Two type 0-variants of Minimalist Grammars". In Jäger, G., P. Monachesi, G. Penn, and S. Wintner, eds. (2002). *Proceedings of Formal Grammar 2002*, Trento.

Lichte, Timm (2007). "An MCTAG with Tuples for coherent constructions in German". In *Proceedings of the 12th Conference on Formal Grammar 2007*. Dublin, Ireland.

Mel'čuk, Igor A. (1974). *Opyt teorii lingvističeskih modelej "Smysl ⇔ Tekst"*. Moscow: Nauka Publishing House. [In Russian: Мельчук И. А. (1974). *Опыт теории лингвистических моделей "Смысл ⇔ Текст"*. Москва: Наука.]

Mel'čuk, Igor A. (1988). *Dependency Syntax: Theory and Practice*. NY: SUNY Publications.

Michaelis, Jens (1998). "Derivational Minimalism is mildly context-sensitive". In Moortgat, M., ed. (1998) *Selected Papers of the 3rd International Conference on Logical Aspects of Computational Linguistics*, LACL'1998, Grenoble, France.

Michaelis, Jens (2001). "Transforming linear context-free rewriting systems into Minimalist Grammars". In de Groote, Ph., G. Morrill and Chr. Retoré, eds. (2001). Proceedings of the 4th International Conference on Logical Aspects of Computational Linguistics, LACL'2001, France.

Michaelis, Jens, and Hans-Martin Gärtner (2003). "A Note on countercyclicity and Minimalist Grammars". In Jäger, G., P. Monachesi, G. Penn and S. Wintner, eds. (2003). *Proceedings of the 8th conference on Formal Grammar*, Vienna, Austria.

Michaelis, Jens (2005). "An additional observation on strict derivational Minimalism". In Jäger, G., P. Monachesi, G. Penn, and S. Wintner, eds. (2002). *Proceedings of Formal Grammar 2002*, Trento.

Perekrestenko, Alexander (2007). "A note on the complexity of the recognition problem for the minimalist grammars with unbounded scrambling and barriers". In Díaz Madrigal, Víctor J., and Fernando Enríquez de Salamanca Ros, eds. (2007). *Actas del XXIII Congreso de la Sociedad Española para el Procesamiento del Lenguaje Natural*, Seville, Spain, 27-34.

Perekrestenko, Alexander (2008). "Minimalist Grammars with unbounded scrambling and nondiscriminating barriers are NP-hard". *Proceedings of the 2nd International Conference on Language and Automata Theory and Applications*, LATA'2008.

Rambow, Owen. 1994. *Formal and Computational Aspects of Natural Language Syntax*. PhD thesis. IRCS Technical Report, University of Pennsylvania.

Søgaard, Anders, Timm Lichte, and Wolfgang Maier (2007). "On the complexity of linguistically motivated extensions of Tree-Adjoining Grammar". In *Recent Advances in Natural Language Processing 2007*. Borovets, Bulgaria.

Stabler, Edward (1997). "Derivational Minimalism". In Retore, Chr., ed. (1997) *Logical Aspects of Computational Linguistics*, Springer, 68-95.

Weir, David J. 1988. *Characterizing mildly context-sensitive formalisms*. PhD thesis. University of Pennsylvania.

8. Appendix 1

Pure tree-adjoining grammar

A pure tree-adjoining grammar (pure TAG) is a 5-tuple $\langle T, N, I, A, S \rangle$ such that:

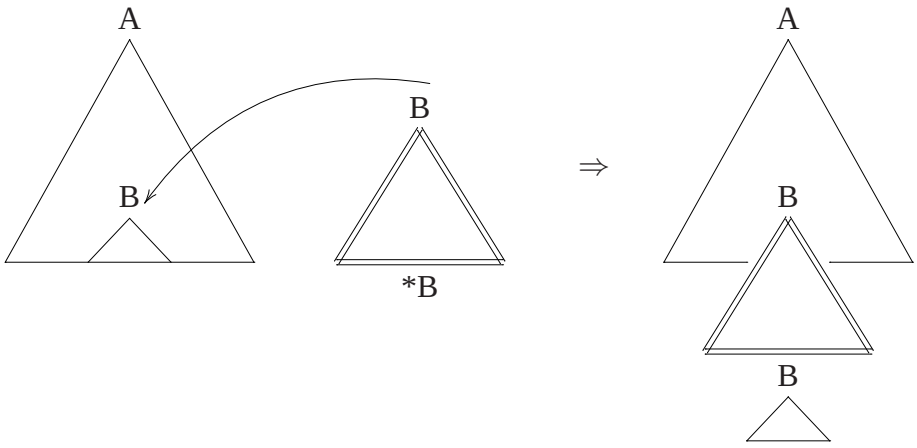
- T is a finite set of *terminal symbols*;
- N is a finite set of *nonterminal symbols*, $T \cap N = \emptyset$;
- S is a distinguished nonterminal symbol (the initial symbol), $S \in N$;
- I is a finite set of *initial trees*, characterized as follows:
  - the interior nodes are labeled by nonterminal symbols,
  - the frontier nodes are labeled by terminals or nonterminals, the nonterminals of the frontier are marked for substitution $(\downarrow)$;
- A is a finite set of *auxiliary trees*, characterized as follows:
  - the interior nodes are labeled by nonterminal symbols,
  - the frontier nodes are labeled by terminals or nonterminals, the nonterminals of the frontier are marked for substitution $(\downarrow)$ except for one nonterminal, which must be label-identical to the root of the tree; this nonterminal is called *foot node* and is marked with an asterisk $(*)$.

The trees in $I \cup A$ are called *elementary trees*. A tree built by their composition is called a *derived tree*. There are two tree-combining operations building new derived trees: adjunction and substitution.

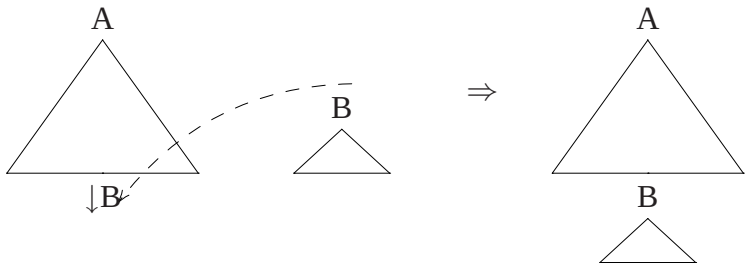
Adjunction builds a new tree inserting a tree that has a foot in place of a non-leaf node of another tree having the same label. Suppose α is a tree containing a non-leaf node n labeled B, and β is a tree containing a foot node also labeled B. The resulting tree, obtained by adjoining β into α, is built as follows:

- the subtree of α with the root n, call it t, is detached leaving a copy of n in α;
- the tree β is attached to the node n of the tree α, and this node n is identified with the root node of β;
- the tree t is attached to the foot node of β, and the root node of t is identified with the foot node of β.

This implies that the tree α, into whose B-labeled node n the three β is adjoined, loses its own B node in that it gets replaced by the three β. By definition, adjunction is disallowed on nodes marked for substitution and normally also in foot nodes. In the illustration below, the B-rooted tree is adjoined in place of the B-labeled node of the tree with the root A:



Substitution builds a new tree by replacing a non-foot leaf node of a tree with another tree without foot. How it works is illustrated below. The *B*-labeled node of the *A*-rooted tree is here replaced with the *B*-rooted tree:



Tree-adjoining grammar

In a pure TAG, a tree β can be adjoined on a non-leaf node *n* of a tree α provided the label of *n* is identical to the root label of β. For the linguistic purposes, it is convenient to have some control over the adjoining operations. This control is realized by means of the so-called *adjunction constraints* which increase the power of the formalism.

A tree-adjoining grammar with adjunction constraints, usually referred to as *tree-adjoining grammar*, is a 5-tuple $\langle T, N, I, A, S\rangle$ such that:

- *T*, *N*, *I*, *A*, and *S* are the same as in the definition of pure TAGs;
- (some of) the internal nodes of the trees are provided with one of the following makers:
  - $OA\{\beta_1, \beta_2, \ldots, \beta_n\}$ – *obligatory adjunction constraint* stipulating that one of the trees $\beta_1, \beta_2, \ldots, \beta_n$ must adjoin at the given node,
  - $SA\{\beta_1, \beta_2, \ldots, \beta_n\}$ – *selective adjunction constraint* allowing only the trees from the set $\{\beta_1, \beta_2, \ldots, \beta_n\}$ to adjoin at the given node,
  - *NA* – *null adjunction constraint* prohibiting any adjunction at the given node.

9. Appendix 2

A Minimalist Grammar with unbounded scrambling and nondiscriminating barriers, MG^{scr}_{B0}, is a tuple $G = \langle \neg Syn, Syn, c, \#, Lex, \Omega \rangle$, such that

- $\neg Syn$ is a finite set of non-syntactic features partitioned into the sets of phonetic (*Phon*) and semantic (*Sem*) features;
- *Syn* is a finite set of syntactic features disjoint from $\neg Syn$ and partitioned into the following sets:
  - base (syntactic) categories, *Base*, partitioned into the set of categories without barrier $B = \{ n, v, d, c, t, \ldots \}$ and the set of categories with barrier $\overline{B} = \{ \overline{n}, \overline{v}, \overline{d}, \overline{c}, \overline{t}, \ldots \}$,
  - m(erge)-selectors $M = \{ =x \mid x \in B \}$,
  - m(ove)-licensees $E = \{ -x \mid x \in B \}$,
  - m(ove)-licensors $R = \{ +x \mid x \in B \}$,
  - s(cramble)-licensees, $S = \{ \sim x \mid x \in B \}$, and
  - '#' which is a special symbol;
- c is a distinguished element of *Base*, the completeness category;
- *Lex* is a *lexicon* defined further on;
- Ω is the set of the structure-building operators '*merge*', '*move*' and '*scramble*' specified later.

Let *Feat* denote the union of the syntactic and non-syntactic features: $Feat = Syn \cup \neg Syn$. An *expression* over the set of features *Feat*, also called a *minimalist tree*, is a five-tuple $\tau = \langle N_\tau, \lhd^*_\tau, \prec_\tau, <_\tau, label_\tau \rangle$, obeying the following conditions:

- $\langle N_\tau, \lhd^*_\tau, \prec_\tau \rangle$ is a finite binary ordered tree, where N_τ is a non-empty finite set of nodes, $\lhd_\tau$ is the binary relation of immediate dominance on N_τ, $\lhd^*_\tau$ is its reflexive transitive closure, and $\prec_\tau$ is the binary relation of precedence on N_τ;
- $<_\tau \subseteq N_\tau \times N_\tau$ is the asymmetric relation of immediate projection that holds for any two sibling nodes, so that for each $x \in N_\tau$ which is not the root of τ either $x <_\tau sibling(x)$ or $sibling(x) <_\tau x$; in the case $x <_\tau y$, we say that x *immediately projects* over y;
- $label_\tau$ is a leaf-labeling function assigning to each leaf of $\langle N_\tau, \lhd^*_\tau, \prec_\tau \rangle$ an element from $Syn^* \{\#\} Syn^* Phon^* Sem^*$, where *Syn*, *Phon*, *Sem* and '#' are as in the definition of MG^{scr}_{B0}.

We denote by $Exp(Feat)$ the set of all expressions over the features *Feat*.

Let $\tau = \langle N_\tau, \lhd^*_\tau, \prec_\tau, <_\tau, label_\tau \rangle \in Exp(Feat)$ be an expression. Leaf $z \in N_\tau$ is the *head* of a given node $x \in N_\tau$ if either z and x are the same node or $x \lhd^+_\tau z$ and for each $y \in N_\tau$, such that $x \lhd^+_\tau y \lhd^*_\tau z$, the following holds: $y <_\tau sibling_\tau(y)$. (Analogously to the notation

$\lhd^*$, we use the shorthand $\lhd^+$ to denote the non-reflexive transitive closure of the dominance relation.) Expression τ is said to be a *head*, or a *simple* expression, if N_τ contains exactly one node. Otherwise τ is said to be a *non-head*, or a *complex* expression. The head of a tree is the head of its root. The root of the tree τ is denoted as r_τ.

Expression corresponding to a given subtree ϕ of τ is referred to as a subexpression of τ. Such subexpression ϕ with root $x \in N_\tau$ is said to be a *maximal projection* in τ if either x is the root of τ (and, as a consequence, does not have any siblings with respect to τ) or ϕ is a proper subexpression of τ and $sibling_\tau(x) <_\tau x$. The set of all maximal projections of τ is denoted as $MaxProj(\tau)$.

Expression $\phi \in MaxProj(\tau)$ over *Feat* is said to *display* feature $f \in Feat$ if its label is in $\alpha\#f\beta$ where $\alpha, \beta \in Feat^*$.

Expression $\phi \in MaxProj(\tau)$ over *Feat* is said to *contain* feature $f \in Feat$ if its label is in $\alpha f\alpha'\#\beta\beta'$ or in $\alpha\alpha'\#\beta f\beta'$ where $\alpha, \alpha', \beta, \beta' \in Feat^*$.

Expression τ is *complete* if its head label is in $Syn^*\{\#\}\{c\}S^?Phon^*Sem^*$ and the labels of all of its leaves are in $Syn^*\{\#\}S^?Phon^*Sem^*$.

Maximal projection τ is *licensed for scrambling* (to x) if the head label of τ displays feature $\sim x$ for some $x \in B$ or $\bar{x} \in \bar{B}$.

Subexpression $\phi \in MaxProj(\tau)$ is *barred for scrambling to* τ if there exists a maximal projection $\chi \in MaxProj(\tau)$ such that $\phi \in MaxProj(\chi)$, $r_\phi \neq r_\chi$, $r_\chi \neq r_\tau$ and the label of χ contains feature $\bar{x} \in \bar{B}$. The head of the subtree χ will be called a *barrier*.

Subexpression $\phi \in MaxProj(\tau)$ is *scrambable to expression* τ if τ is a maximal projection, for some $x \in Base$ the head label of τ displays category x or $\bar{x}$, the subexpression ϕ is licensed for scrambling to x and is not barred for scrambling to τ. For a given maximal projection τ, the set of all expressions scrambable to it will be denoted as $ScrS(\tau)$.

For two expressions $\phi, \chi \in Exp(Feat)$, $[<\phi,\chi]$ (respectively, $[>\phi,\chi]$) denotes the complex expression $\psi = \langle N_\psi, \lhd^*_\psi, \prec_\psi, <_\psi, label_\psi\rangle \in Exp(Feat)$ such that $r_\psi \lhd_\psi r_\phi$, $r_\psi \lhd_\psi r_\chi$, $r_\phi \prec_\psi r_\chi$, and $r_\phi <_\psi r_\chi$ (respectively, $r_\chi <_\psi r_\phi$).

The phonetic yield of the (complex) expression τ, $Y_{Phon}(\tau)$, is defined as the concatenation of the *Phon* string of the leaf labels in the order in which they occur in the tree τ.

The lexicon of MG^{scr}_{B0}, *Lex*, is a finite set of simple expressions over *Feat*, each of which is of the form $\phi = \langle N_\phi, \lhd^*_\phi, \prec_\phi, <_\phi, label_\phi\rangle$ with $N_\phi = \{\varepsilon\}$ and the leaf-labeling function $label_\phi$ assigns to the only node of ϕ an element from

$$\{\#\}\, M^*\, R^*\, Base\, (E \cup S)^?\, Phon^*\, Sem^*.$$

The structure-building operators Ω of MG^{scr}_{B0} are defined below in terms of their mapping type, domain and operation.

Operator *merge*

Type: partial mapping $Exp(Feat) \times Exp(Feat) \longrightarrow Exp(Feat)$.

Domain: for any $\phi, \chi \in Exp(Feat)$, the tuple $\langle\phi,\chi\rangle$ is in $Dom(merge)$ iff for some $x \in B$ the head label of ϕ displays m-selector $=x$ and the head label of χ displays category x or $\bar{x}$.

Operation:

$$merge(\phi,\chi) = \begin{cases} [<\phi',\chi'] & \mid \phi \text{ is simple} \\ [>\chi',\phi'] & \mid \phi \text{ is complex} \end{cases}$$

where ϕ' and χ' result from the corresponding ϕ and χ by swapping the # symbol with the feature immediately following it to the right.

Operator *move*

Type: partial mapping $Exp(Feat) \longrightarrow Exp(Feat)$.

Domain: for any $\phi \in Exp(Feat)$, the expression ϕ is in $Dom(move)$ iff for some $x \in B$ the head label of ϕ displays m-licensor $+x$ and there is a unique $\chi \in MaxProj(\phi)$ displaying m-licensee $-x$. The uniqueness of χ prohibiting the occurrence of two or more competing movement candidates is the way the SMC is implemented for the *move* operator.

Operation: $move(\phi) = [_{>} \chi', \phi']$, where ϕ' and χ' result from the corresponding ϕ and χ by swapping the # symbol with the feature immediately following it to the right and replacing the subtree χ in ϕ with a single (empty) node labeled ε.

Operator *scramble*

Type: partial mapping $Exp(Feat) \longrightarrow 2^{Exp(Feat)}$.

Domain: for any $\phi \in Exp(Feat)$, the expression ϕ is in $Dom(scramble)$ if and only if $ScrS(\phi) \neq \emptyset$.

For a given set of expressions $\Phi \subseteq Exp(Feat)$, let
$S'(\Phi) = \{\, [_{>} \chi', \phi'] \mid \phi \in \Phi,\ \phi \in Dom(scramble),\ \chi \in ScrS(\phi),\ \phi'$ results from ϕ by replacing subtree χ by a single empty node labeled ε, and χ' results from χ by swapping the # symbol with the feature immediately following it to the right$\,\}$.
Let $S^0(\Phi) = \Phi$ and $S^{k+1}(\Phi) = S^k(\Phi) \cup S'(S^k(\Phi))$ for $k \geq 0$, $k \in \mathbb{N}$.

Operation: $scramble(\phi) = \bigcup_{k \in \mathbb{N}} S^k(\{\phi\})$.

Let $G = \langle \neg Syn, Syn, c, \#, Lex, \Omega \rangle$ be an MG^{scr}_{B0}. Let $CL^0(G) = Lex$. For $k > 0$, $k \in \mathbb{N}$, $CL^k(G)$ will be defined as follows:

$$\begin{aligned} CL^{k+1}(G) = CL^k(G) \cup \quad & \{\, merge(\phi, \chi) \mid \phi, \chi \in CL^k(G) \,\} \cup \\ & \{\, move(\phi) \mid \phi \in CL^k(G) \,\} \cup \\ & \{\, scramble(\phi) \mid \phi \in CL^k(G) \,\} \end{aligned}$$

Let $CL(G) = \bigcup_{k \in \mathbb{N}} CL^k(G)$. The tree language of G, $MT^{scr}_{B0}(G)$, is defined in the following way:

$$MT^{scr}_{B0}(G) \;=\; \{\, \tau \mid \tau \in CL(G) \text{ and } \tau \text{ is complete} \,\}.$$

The string language of G, $ML^{scr}_{B0}(G)$, is defined as the yields of its tree language:

$$ML^{scr}_{B0}(G) \;=\; \{\, Y_{Phon}(\tau) \mid \tau \in MT^{scr}_{B0}(G) \,\}.$$