Computational complexity of commutative grammars

Jérôme Kirman Sylvain Salvati

Bordeaux I University - LaBRI
INRIA

October 1, 2013

Free word order

Some languages allow free placement of several words or phrases:

- Fully non-configurational languages (Latin)
- Scrambling phenomenon (German)

Case marking or pragmatics give information on syntactic roles

Grammatically, all the possible word orders are equally valid, while the sentence's meaning and syntactic roles stay the same.

Scrambling in German subordinate clauses [BNR92]

	1	2	3	4	5	
...daß	eine hiesige Firma	meinem Onkel	die Möbel	vor drei Tagen	ohne Voranmeldung	zugestellt hat.
...that	a local company	to my uncle	the furniture	three days ago	without warning	has delivered.

...that a local company has delivered the furniture to my uncle three days ago without warning.

All the possible orderings of the constituents (1,2,3,4,5) are possible

Arguments of nested subordinate clauses can also be scrambled

Words modulo commutation

There is a single sentence, which has many representations.

One structure $\longleftrightarrow$ Many strings

Words modulo commutation

There is a single sentence, which has many representations.

One structure $\longleftrightarrow$ Many strings

$\Rightarrow$ We need an object that represents a word modulo commutation of some of its letters/factors (sub-sequences) and a way to parse it

An algebra for words modulo commutation

Words modulo commutation over an alphabet Σ will be constructed as terms of an algebra $\mathrm{com}(\Sigma)$

Atomic terms are letters of the alphabet Σ and the empty word ε

Terms are constructed by means of two operators:

- $\bullet$ Concatenation of subterms (associative, as usual)
- $\otimes$ Commutative combination of subterms (assoc + comm)

Terms and associated languages

Ranked signature:

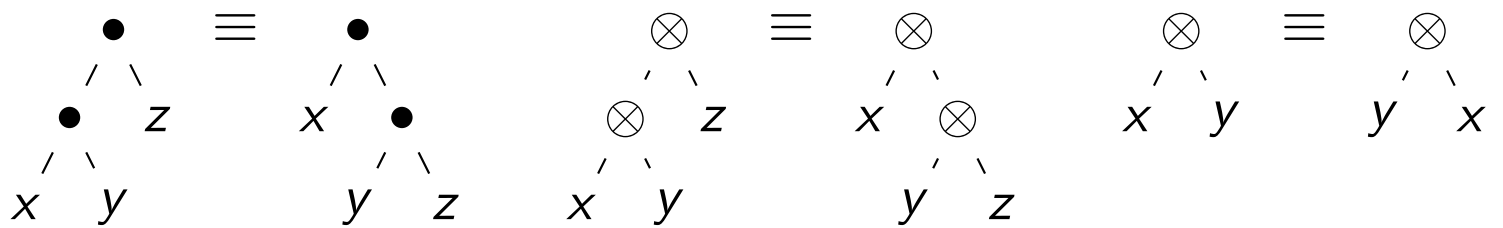
$$\mathrm{com}(\Sigma) := \{a^0 | a \in \Sigma\} \cup \{\varepsilon^0; \bullet^2; \otimes^2\}$$

Terms and associated languages

Ranked signature:

$$\mathrm{com}(\Sigma) := \{a^0 | a \in \Sigma\} \cup \{\varepsilon^0; \bullet^2; \otimes^2\}$$

Equivalence relation for terms ($\equiv$):



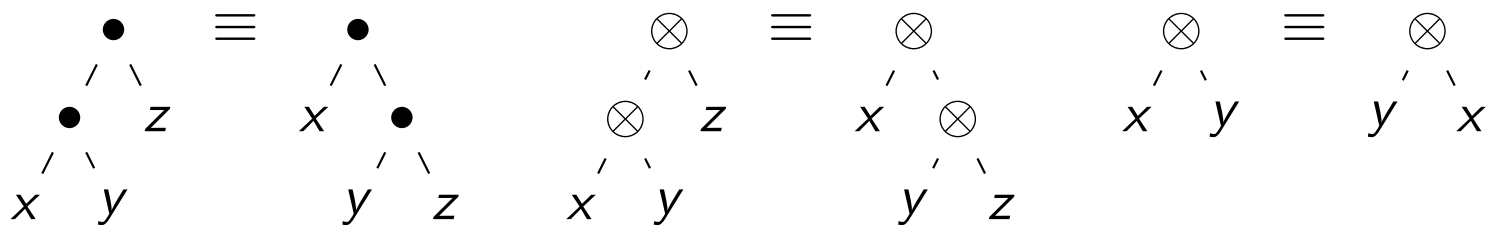
• associativity ⊗ associativity ⊗ commutativity

Terms and associated languages

Ranked signature:

$$\mathrm{com}(\Sigma) := \{a^0 | a \in \Sigma\} \cup \{\varepsilon^0; \bullet^2; \otimes^2\}$$

Equivalence relation for terms ($\equiv$):

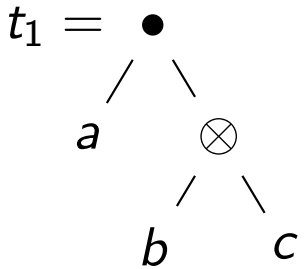


• associativity ⊗ associativity ⊗ commutativity

Language of a term:

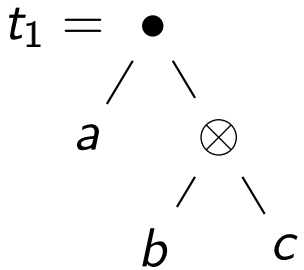
$$L(t) := \{yield(t') | t' \equiv t\}$$

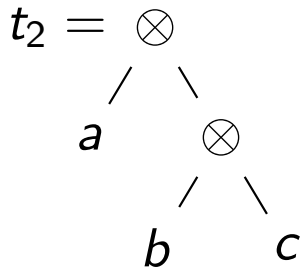
Examples of terms



- $L(t_1) = \{abc, acb\}$

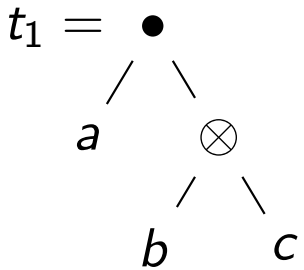
Examples of terms

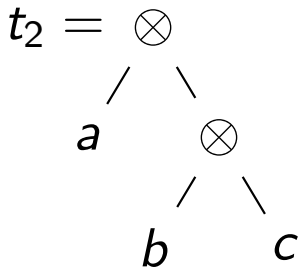


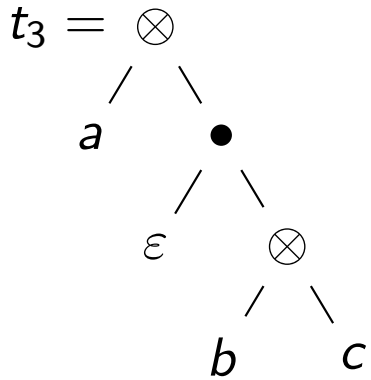


- $L(t_1) = \{abc, acb\}$
- $L(t_2) = \{abc, acb, bac, bca, cab, cba\}$

Examples of terms



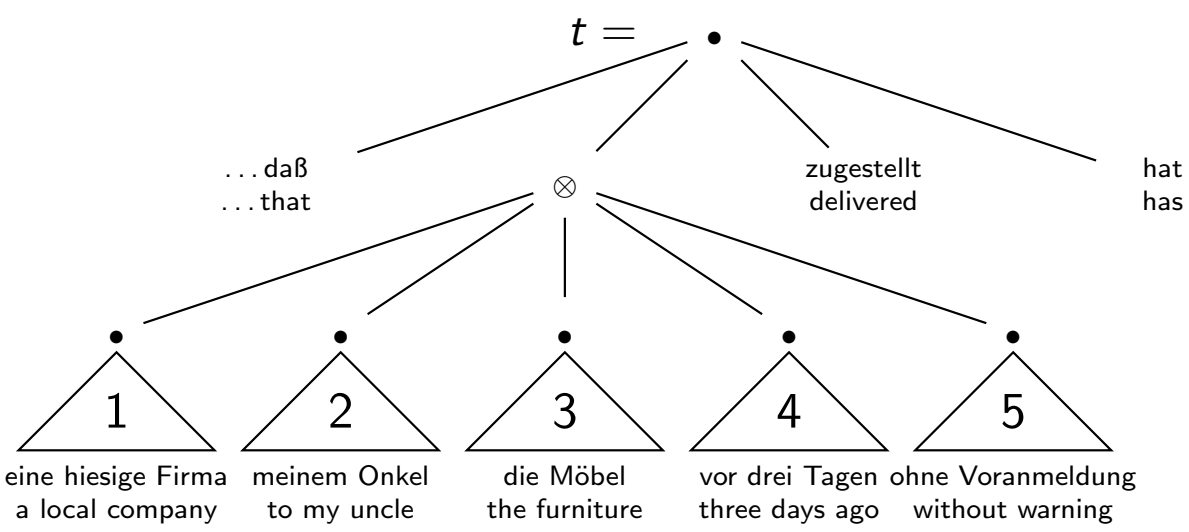




- $L(t_1) = \{abc, acb\}$
- $L(t_2) = \{abc, acb, bac, bca, cab, cba\}$
- $L(t_3) = \{abc, acb, bca, cba\}$

Packed representation

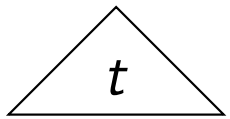
Using this algebra, we get objects to represent free word orders:



$$L(t) = \left\{ \begin{array}{c} \text{daß 1 2 3 4 5 zugestellt hat} \\ \text{daß 1 2 3 5 4 zugestellt hat} \\ \text{daß 1 2 4 3 5 zugestellt hat} \\ \vdots \\ \text{daß 2 5 4 3 1 zugestellt hat} \\ \text{daß 3 1 2 4 5 zugestellt hat} \\ \vdots \\ \text{daß 4 5 3 2 1 zugestellt hat} \\ \text{daß 5 1 2 3 4 zugestellt hat} \\ \vdots \\ \text{daß 5 4 3 1 2 zugestellt hat} \\ \text{daß 5 4 3 2 1 zugestellt hat} \end{array} \right\}$$

$5! = 120$ word orders

Towards commutative grammars



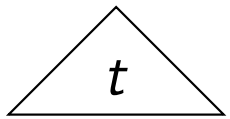
sentence with free order

$\leftrightarrow$

$\{w \mid w \in L(t)\}$

set of representations

Towards commutative grammars

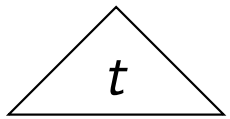


sentence with free order $\leftrightarrow$ $\{w \mid w \in L(t)\}$ set of representations

Commutative grammars are defined as term grammars, with:

$$L_w(G) := \bigcup_{t \in L(G)} L(t)$$

Towards commutative grammars



$\leftrightarrow \quad \{w \mid w \in L(t)\}$

sentence with free order — set of representations

Commutative grammars are defined as term grammars, with:

$$L_w(G) := \bigcup_{t \in L(G)} L(t)$$

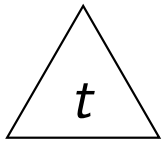
$$L(G) = \left\{ \triangle\, t_1 \;;\; \triangle\, t_2 \;;\ldots \right\} \longrightarrow L_w(G) = L(t_1) \cup L(t_2) \cup \ldots$$

Commutative context-free grammars

The first natural class of grammars we consider is **CCFG**:

- Regular term grammars on the terminal signature $\mathrm{com}(\Sigma)$
- **CCFG** generalize CFG (without the $\otimes$ operator)

$$\mathcal{G} = \langle \mathcal{N}, \mathrm{com}(\Sigma), \mathcal{R}, S \rangle$$

The rules of $\mathcal{R}$ have the form $N \to$ 

, with $t \in \mathrm{com}(\mathcal{N} \cup \Sigma)$

Example of **CCFG**

$$G := \langle \{S, W\}, \mathrm{com}(\{a, b, c, \#\}), \mathcal{R}, S \rangle$$

$$\mathcal{R} := \left\{ \begin{array}{l} S \longrightarrow \bullet \quad ; S \to \varepsilon \,; \quad W \longrightarrow \otimes \; ; W \to \varepsilon \\ \qquad\quad / \quad \backslash \qquad\qquad\qquad\quad / \quad \backslash \\ \qquad \bullet \quad S \qquad\qquad\qquad \otimes \quad\; \otimes \\ \qquad / \quad \backslash \qquad\qquad\qquad / \; \backslash \quad / \; \backslash \\ W \quad \# \qquad\qquad\qquad a \quad b \quad c \quad W \end{array} \right\}$$

Example of **CCFG**

$$G := \langle \{S, W\}, \mathrm{com}(\{a, b, c, \#\}), \mathcal{R}, S \rangle$$

$$\mathcal{R} := \left\{ \begin{array}{l} S \longrightarrow \bullet \quad ; S \to \varepsilon \,; \quad W \longrightarrow \otimes \; ; W \to \varepsilon \\ \qquad\quad / \quad \backslash \qquad\qquad\qquad\quad / \quad \backslash \\ \qquad \bullet \quad S \qquad\qquad\qquad \otimes \quad\; \otimes \\ \qquad / \quad \backslash \qquad\qquad\qquad / \; \backslash \quad / \; \backslash \\ W \quad \# \qquad\qquad\qquad a \quad b \quad c \quad W \end{array} \right\}$$

$$L_w(G) = \{w_1 \# \dots w_n \# | n \in \mathbb{N} \wedge |w_i|_a = |w_i|_b = |w_i|_c\}$$

Example of **CCFG**

$$G := \langle \{S, W\}, \mathrm{com}(\{a, b, c, \#\}), \mathcal{R}, S \rangle$$

$$\mathcal{R} := \left\{ S \longrightarrow \begin{array}{c} \bullet \\ / \quad \backslash \\ \begin{array}{c} \bullet \\ / \ \backslash \\ W \quad \# \end{array} \quad S \end{array} ; S \to \varepsilon ; \quad W \longrightarrow \begin{array}{c} \otimes \\ / \quad \backslash \\ \begin{array}{c} \otimes \\ / \ \backslash \\ a \quad b \end{array} \quad \begin{array}{c} \otimes \\ / \ \backslash \\ c \quad W \end{array} \end{array} ; W \to \varepsilon \right\}$$

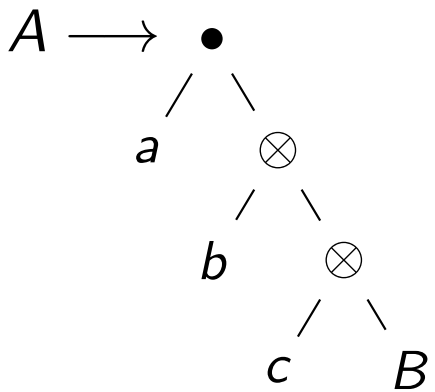
$$L_w(G) = \{ w_1\# \ldots w_n\# | n \in \mathbb{N} \wedge |w_i|_a = |w_i|_b = |w_i|_c \}$$

$$\mathcal{R} = \left\{ \begin{array}{l} S \left(\begin{array}{c} \bullet \\ / \quad \backslash \\ \begin{array}{c} \bullet \\ / \ \backslash \\ x_2 \quad \# \end{array} \quad x_1 \end{array} \right) \leftarrow S(x_1)\ W(x_2) \ ; \ W \left(\begin{array}{c} \otimes \\ / \quad \backslash \\ \begin{array}{c} \otimes \\ / \ \backslash \\ a \quad b \end{array} \quad \begin{array}{c} \otimes \\ / \ \backslash \\ c \quad x_1 \end{array} \end{array} \right) \leftarrow W(x_1) \\ S(\varepsilon) \longleftarrow \qquad ; \ W(\varepsilon) \longleftarrow \end{array} \right\}$$

Commutative regular grammars

CREG are a more constrained version of **CCFG**:

- Right-hand sides are restricted to right-branching terms with a single non-terminal at the end (rightmost leaf)
- **CREG** generalize right-linear (regular) word grammars



A typical **CREG** production

Commutative multiple regular grammar

CMREG extend **CCFG** by allowing multiple terms:

- Productions have the form:
  $$A(t_1, \ldots, t_n) \leftarrow B_1(x_{1,1}, \ldots, x_{1,n_1}), \ldots, B_p(x_{p,1}, \ldots, x_{p,n_p})$$
  Where $t_i \in \mathrm{com}\left(\Sigma \cup \left\{ x_{i,j} \mid \begin{smallmatrix} 1 \leq i \leq p \\ 1 \leq j \leq n_i \end{smallmatrix} \right\}\right)$.
- Nonterminals are typed for consistency, with $\tau(S) = [o]$.
- **CMREG** generalize multiple context free word grammars.

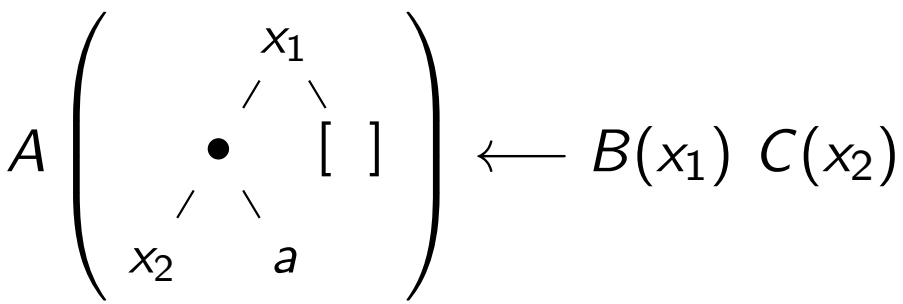
$$A\left(\begin{array}{ccc} & \otimes & \\ & / \quad \backslash & \\ x_{1,1} & & x_{2,1} \end{array}, \begin{array}{ccc} & \otimes & \\ & / \quad \backslash & \\ x_{1,2} & & x_{2,2} \end{array}\right) \longleftarrow B(x_{1,1}, x_{1,2})\ C(x_{2,1}, x_{2,2})$$

Example of a **CMREG** production.

Commutative macro grammars

CMG are another generalization of **CCFG**:

- Nonterminals construct contexts (not just ground terms).
- Nonterminals are typed to enforce arities, with $\tau(S) = [o]$.
- **CMG** generalize well-nested MCFGs.

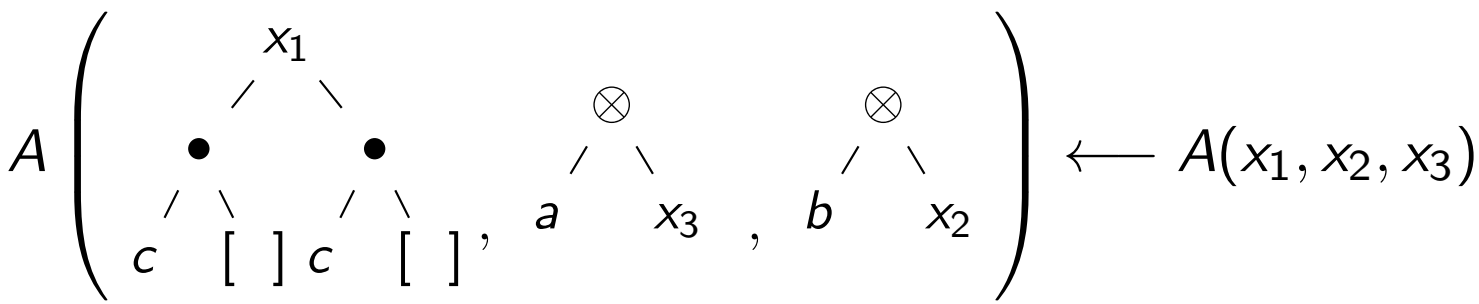


Example of a **CMG** production.

Commutative multiple context-free grammars

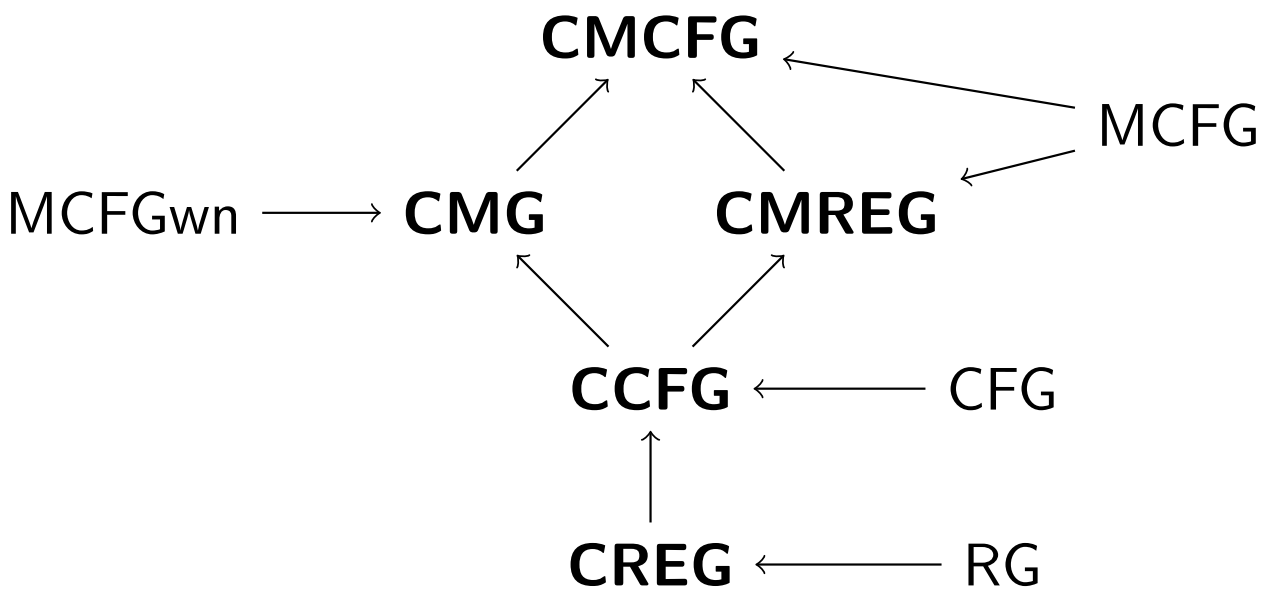
CMCFG generalize the other classes:

- Nonterminals construct tuples of contexts.
- Intuitively the "meet" of **CMG** and **CMREG**.
- Nonterminals are also typed with $\tau(S) = [o]$.



Example of a **CMCFG** production.

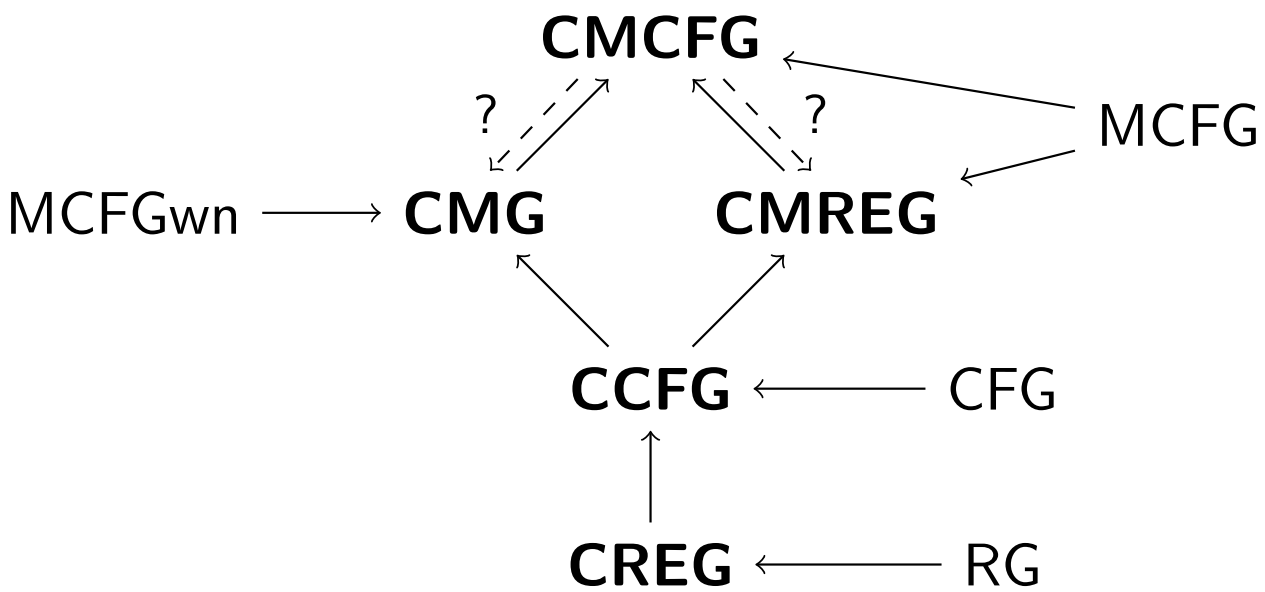
Outline of commutative grammar hierarchy



Hierarchy of mildly context-sensitive commutative grammars

(arrows denote inclusion)

Outline of commutative grammar hierarchy



Hierarchy of mildly context-sensitive commutative grammars

(arrows denote inclusion)

Two decisions problems for parsing

We consider two classes of decision problems related to parsing:

- Universal membership problem

  Input A word w and a grammar G.
  Answer *YES* iff $w \in L_w(G)$.

- Membership problem for G

  Input A word w.
  Answer *YES* iff $w \in L_w(G)$.

Solving both problems involves parsing w according to G; however, for membership, G is a fixed-size parameter (part of the constant).

Overview of results

Class	Membership	Universal membership	
		ND	Full
Terms	O(1)	NP-C	
CREG	NLOGSPACE	NP-C	
CCFG	LOGCFL-C	NP-C	
CMG	NP-C	PSPACE-C	EXPTIME
CMREG	NP-C	PSPACE-C	EXPTIME-C
CMCFG	NP-C	PSPACE-C	EXPTIME-C

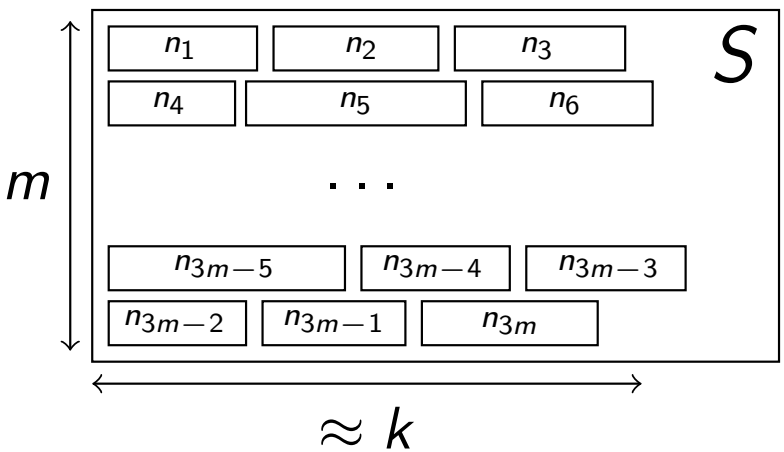
NP-completeness of universal term membership

Class	Membership	Universal membership	
		ND	Full
Terms	O(1)	NP-C	
CREG	NLOGSPACE	NP-C	
CCFG	LOGCFL-C	NP-C	
CMG	NP-C	PSPACE-C	EXPTIME
CMREG	NP-C	PSPACE-C	EXPTIME-C
CMCFG	NP-C	PSPACE-C	EXPTIME-C

NP-hardness of UTM (1/2)

3-PART problem:

Input A set S of $3m$ integers $\left(\frac{k}{4} \leq n_i \leq \frac{k}{2}\right)$



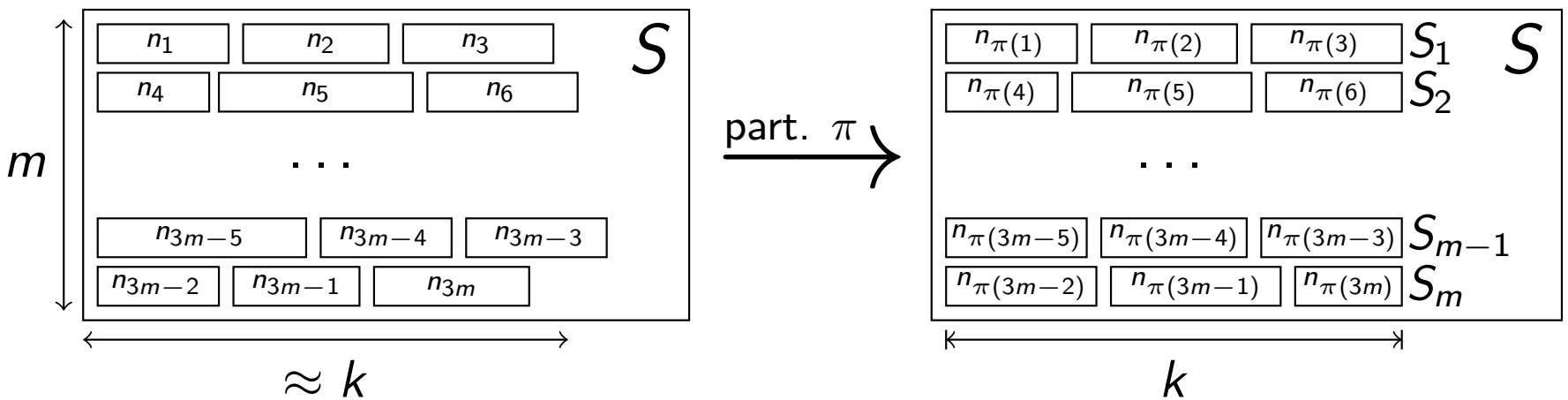
NP-hardness of UTM (1/2)

3-PART problem:

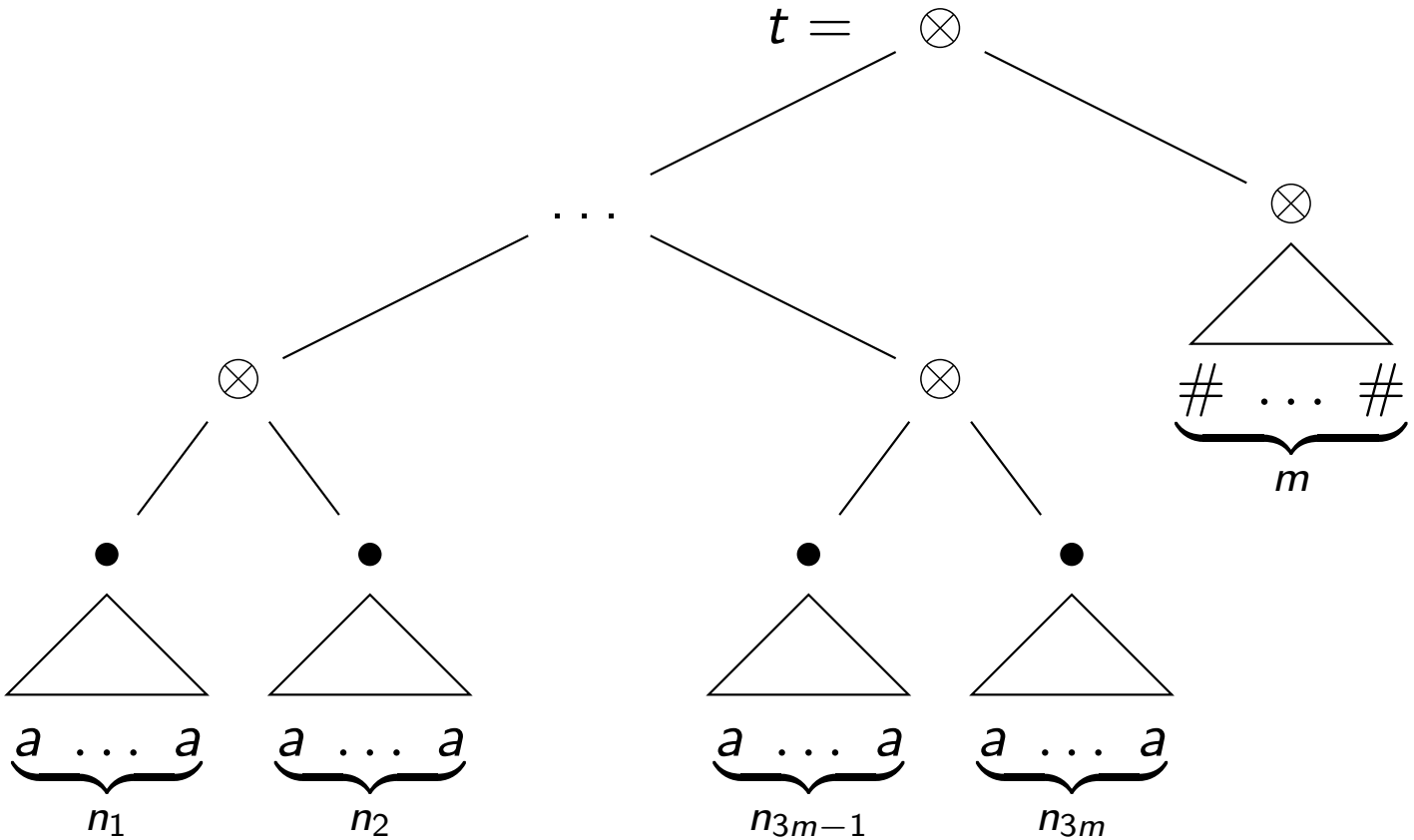
Input A set S of $3m$ integers $\left(\frac{k}{4} \leq n_i \leq \frac{k}{2}\right)$

Answer *YES* iff there is a partition $S_1 \dots S_m$ of S that satisfies:

$$\bigwedge_{S_i} \left(\sum_{n_j \in S_i} n_j \right) = k$$



NP-hardness of UTM (2/2)



$$w = (a^k \#)^m$$

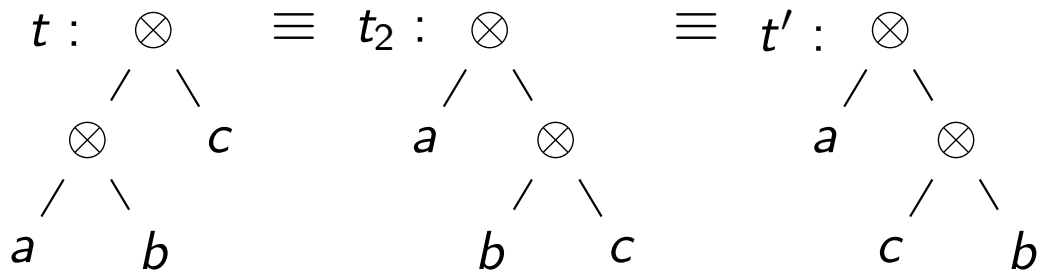
$$w = \underbrace{\underbrace{a \ \ldots \ a}_{k}\#\underbrace{a \ \ldots \ a}_{k}\# \quad \ldots \quad \underbrace{a \ \ldots \ a}_{k}\#}_{m}$$

Outline of NP algorithm for UTM

To check wether $w \in L(t)$:

- Guess a term t' (s.t. $t' \equiv t$ in $\mathrm{com}(\Sigma)$ by assoc/comm of $\otimes$)
- Check that $\mathit{yield}(t') = w$
- Check that $t' \equiv t$

$$w = acb \in L(t)?$$



$$t \equiv t' \wedge \mathit{yield}(t') = acb \Rightarrow acb \in L(t)$$

NP-completeness of universal **CCFG** membership

Class	Membership	Universal membership	
		ND	Full
Terms	O(1)	NP-C	
CREG	NLOGSPACE	NP-C	
CCFG	LOGCFL-C	NP-C	
CMG	NP-C	PSPACE-C	EXPTIME
CMREG	NP-C	PSPACE-C	EXPTIME-C
CMCFG	NP-C	PSPACE-C	EXPTIME-C

Outline of NP algorithm for **CCFG**

Deciding wether $w \in L_w(G)$:

- Construct a compact grammar G' s.t. $L_w(G') = L_w(G)$.
- Guess a derivation of a term $t \in L(G')$ with $|t| \leq |w|^k$.
- Check that the derivation is valid in PTIME.
- Check that $w \in L(t)$ in NP.

Why is there such a G' ?

$$\otimes(t, \varepsilon) \to_\varepsilon t$$
$$\otimes(\varepsilon, t) \to_\varepsilon t$$

$$\left.\begin{array}{l} \bullet(t, \varepsilon) \to_\varepsilon t \\ \bullet(\varepsilon, t) \to_\varepsilon t \end{array}\right\} \text{ iff } |t| = 1 \text{ or } t = \bullet(t_1, t_2)$$

Language-preserving rewriting system for terms.

LOGCFL-completeness of **CCFG** membership

Class	Membership	Universal membership	
		ND	Full
Terms	O(1)	NP-C	
CREG	NLOGSPACE	NP-C	
CCFG	LOGCFL-C	NP-C	
CMG	NP-C	PSPACE-C	EXPTIME
CMREG	NP-C	PSPACE-C	EXPTIME-C
CMCFG	NP-C	PSPACE-C	EXPTIME-C

Notations for the **CCFG** algorithm

Grammars are "compact" and in normal form:

$$A\left(\begin{array}{ccc} & [\mathrm{op}] & \\ \swarrow & & \searrow \\ x & & y \end{array}\right) \longleftarrow B(x)\ C(y)$$

$$A(a) \longleftarrow$$

Derivation items are unordered vectors of NT between two positions:

$$\langle \vec{v}, i, j \rangle \qquad (1 \leq i, j \leq |w|)$$

$\psi(G, N)$ is the precomputed (semilinear) set of bags of non-terminals that N can generate in any order according to G.

CCFG parsing algorithm

$$\frac{A(a) \quad a = w[i,j]}{\langle 1_A, i, j\rangle}\ \textsc{constant}$$

$$\frac{\langle 1_B, i, j\rangle \quad \langle 1_C, j, k\rangle \quad A(\bullet x\, y) \leftarrow B(x)\ C(y)}{\langle 1_A, i, k\rangle}\ \textsc{combine}$$

$$\frac{\langle v_1, i, j\rangle \quad \langle v_2, j, k\rangle \quad 0 < v_1 \quad 0 < v_2 \quad |v_1 + v_2| \leq |w|}{\langle v_1 + v_2, i, k\rangle}\ \textsc{comm. combine}$$

$$\frac{\langle v, i, j\rangle \quad v \in \psi(G, A)}{\langle 1_A, i, j\rangle}\ \textsc{comm. reduction}$$

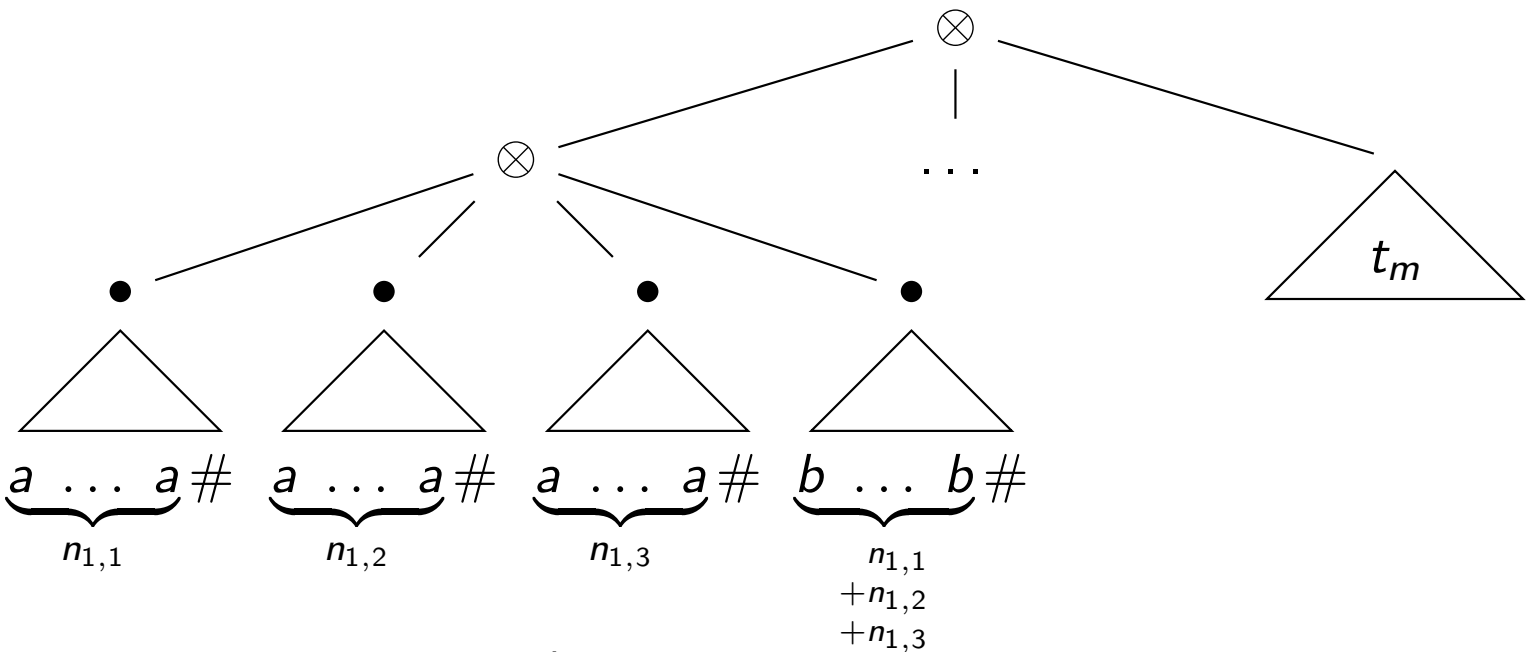
LOGCFL recognition algorithm for **CCFG**.

NP-hardness of fixed grammars beyond **CCFG**

Class	Membership	Universal membership	
		ND	Full
Terms	O(1)	NP-C	
CREG	NLOGSPACE	NP-C	
CCFG	LOGCFL-C	NP-C	
CMG	NP-C	PSPACE-C	EXPTIME
CMREG	NP-C	PSPACE-C	EXPTIME-C
CMCFG	NP-C	PSPACE-C	EXPTIME-C

Yet another reduction to **3-PART**

Based on **3-PART** - triplets and their sum are derived in parallel.



$$w = a^{n_1}\# \ldots a^{n_{3m}}\#(b^k\#)^m$$

$$w = \underbrace{a \ldots a}_{n_1}\# \ldots \underbrace{a \ldots a}_{n_{3m}}\# \underbrace{\underbrace{b \ldots b}_{k}\# \ldots \underbrace{b \ldots b}_{k}\#}_{m}$$

NP-hard fixed **CMREG**

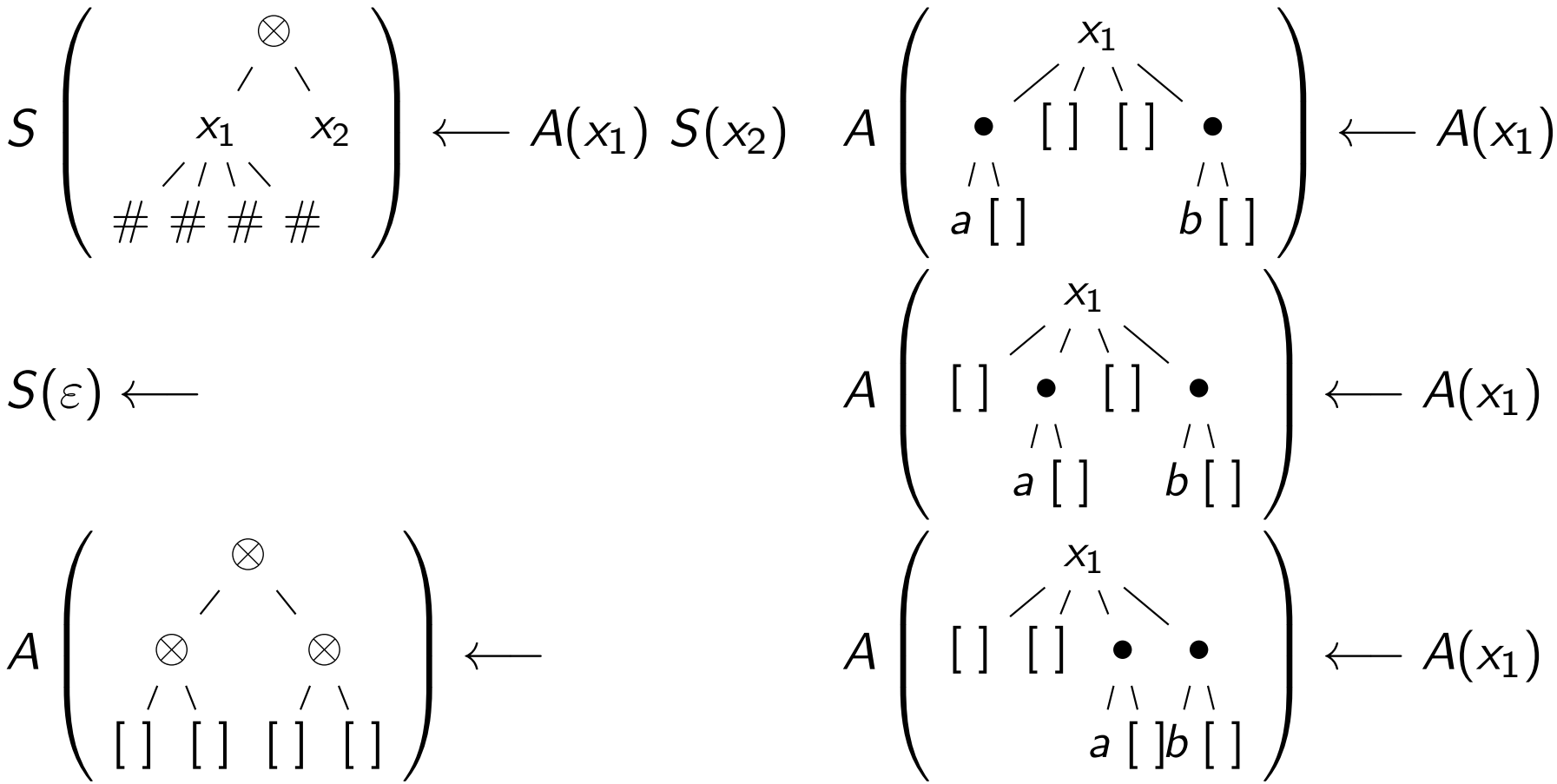
$$S\left(\begin{array}{c} \otimes \\ / \quad \backslash \\ \otimes \qquad x_5 \\ / \quad \backslash \\ \otimes \qquad \otimes \\ / \ \backslash \quad / \ \backslash \\ x_1 \ x_2 \ x_3 \ x_4 \end{array}\right) \longleftarrow A(x_1)\ S(x_2) \qquad S(\varepsilon) \longleftarrow$$

$$A\left(\begin{array}{c} \bullet \\ / \ \backslash \\ a \ x_1 \end{array}, x_2, x_3, \begin{array}{c} \bullet \\ / \ \backslash \\ b \ x_4 \end{array}\right) \longleftarrow A(x_1, x_2, x_3, x_4) \qquad A(\#, \#, \#, \#) \longleftarrow$$

$$A\left(x_1, \begin{array}{c} \bullet \\ / \ \backslash \\ a \ x_2 \end{array}, x_3, \begin{array}{c} \bullet \\ / \ \backslash \\ b \ x_4 \end{array}\right) \longleftarrow A(x_1, x_2, x_3, x_4)$$

$$A\left(x_1, x_2, \begin{array}{c} \bullet \\ / \ \backslash \\ a \ x_3 \end{array}, \begin{array}{c} \bullet \\ / \ \backslash \\ b \ x_4 \end{array}\right) \longleftarrow A(x_1, x_2, x_3, x_4)$$

NP-hard fixed **CMG**



Overview of results

Class	Membership	Universal membership	
		ND	Full
Terms	O(1)	NP-C	
CREG	NLOGSPACE	NP-C	
CCFG	LOGCFL-C	NP-C	
CMG	NP-C	PSPACE-C	EXPTIME
CMREG	NP-C	PSPACE-C	EXPTIME-C
CMCFG	NP-C	PSPACE-C	EXPTIME-C

Conclusion and perspectives

In summary, we have provided:

- An algebraic representation of sentences with free order
- A hierarchy of generative grammars to construct such representations
- A study of the associated computational complexities

Next we want to look into:

- Closure properties : rational cones, AFL, ...
- Relation with UVG, dependency parsing, ...