On the Computational Complexity of Dominance Links in Grammatical Formalisms

Sylvain Schmitz

LSV, ENS Cachan & CNRS

LIFO Seminars, June 28, 2010

Linguistic Motivations

Computational Motivations

Complexity Survey

References

Scrambling

(Becker et al., 1991; Rambow, 1994a; Lichte, 2007, . . .)

dass	Peter	den Kühlschrank	zu versuchen	zu reparieren	verspricht
that	Peter$_{nom}$	the fridge$_{acc}$	to try	to repair	promises

dass	den Kühlschrank	Peter	zu versuchen	zu reparieren	verspricht
that	the fridge$_{acc}$	Peter$_{nom}$	to try	to repair	promises

dass	den Kühlschrank	Peter	zu reparieren	zu versuchen	verspricht
that	the fridge$_{acc}$	Peter$_{nom}$	to repair	to try	promises

dass	den Kühlschrank	zu reparieren	Peter	zu versuchen	verspricht
that	the fridge$_{acc}$	to repair	Peter$_{nom}$	to try	promises

*	dass	den Kühlschrank	zu versuchen	Peter	zu reparieren	verspricht
	that	the fridge$_{acc}$	to try	Peter$_{nom}$	to repair	promises

Scrambling

(Becker et al., 1991; Rambow, 1994a; Lichte, 2007, . . .)

	dass	[Peter]	[den Kühlschrank]	zu versuchen	zu reparieren	verspricht
	that	Peter_{nom}	the fridge_{acc}	to try	to repair	promises
	dass	[den Kühlschrank]	[Peter]	zu versuchen	zu reparieren	verspricht
	that	the fridge_{acc}	Peter_{nom}	to try	to repair	promises
	dass	den Kühlschrank	Peter	zu reparieren	zu versuchen	verspricht
	that	the fridge_{acc}	Peter_{nom}	to repair	to try	promises
	dass	den Kühlschrank	zu reparieren	Peter	zu versuchen	verspricht
	that	the fridge_{acc}	to repair	Peter_{nom}	to try	promises
*	dass	den Kühlschrank	zu versuchen	Peter	zu reparieren	verspricht
	that	the fridge_{acc}	to try	Peter_{nom}	to repair	promises

Scrambling

(Becker et al., 1991; Rambow, 1994a; Lichte, 2007, . . .)

dass	Peter	den Kühlschrank	zu versuchen	zu reparieren	verspricht
that	Peter$_{nom}$	the fridge$_{acc}$	to try	to repair	promises

dass	den Kühlschrank	Peter	[zu versuchen]	[zu reparieren]	verspricht
that	the fridge$_{acc}$	Peter$_{nom}$	to try	to repair	promises

dass	den Kühlschrank	Peter	[zu reparieren]	[zu versuchen]	verspricht
that	the fridge$_{acc}$	Peter$_{nom}$	to repair	to try	promises

dass	den Kühlschrank	zu reparieren	Peter	zu versuchen	verspricht
that	the fridge$_{acc}$	to repair	Peter$_{nom}$	to try	promises

*	dass	den Kühlschrank	zu versuchen	Peter	zu reparieren	verspricht
	that	the fridge$_{acc}$	to try	Peter$_{nom}$	to repair	promises

Scrambling

(Becker et al., 1991; Rambow, 1994a; Lichte, 2007, . . .)

dass	Peter	den Kühlschrank	zu versuchen	zu reparieren	verspricht
that	Peter_{nom}	the fridge_{acc}	to try	to repair	promises

dass	den Kühlschrank	Peter	zu versuchen	zu reparieren	verspricht
that	the fridge_{acc}	Peter_{nom}	to try	to repair	promises

dass	den Kühlschrank	[Peter]	[zu reparieren]	zu versuchen	verspricht
that	the fridge_{acc}	Peter_{nom}	to repair	to try	promises

dass	den Kühlschrank	[zu reparieren]	[Peter]	zu versuchen	verspricht
that	the fridge_{acc}	to repair	Peter_{nom}	to try	promises

*	dass	den Kühlschrank	zu versuchen	Peter	zu reparieren	verspricht
	that	the fridge_{acc}	to try	Peter_{nom}	to repair	promises

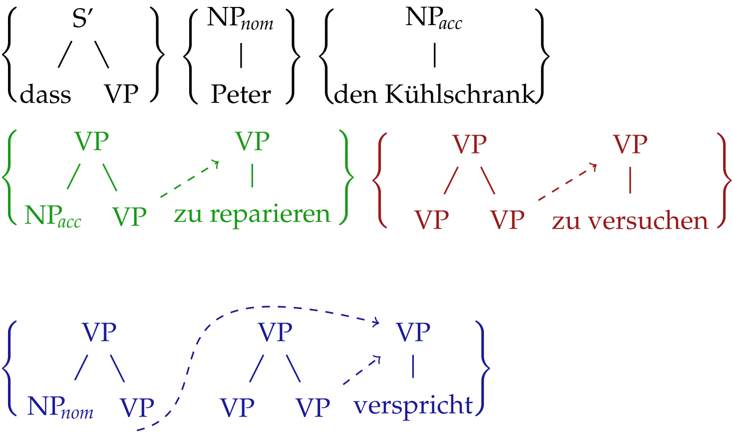
Scrambling

(Becker et al., 1991; Rambow, 1994a; Lichte, 2007, . . .)

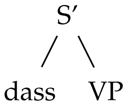
	dass	Peter	den Kühlschrank	zu versuchen	zu reparieren	verspricht
	that	Peter_{nom}	the fridge_{acc}	to try	to repair	promises
	dass	den Kühlschrank	Peter	zu versuchen	zu reparieren	verspricht
	that	the fridge_{acc}	Peter_{nom}	to try	to repair	promises
	dass	den Kühlschrank	Peter	zu reparieren	zu versuchen	verspricht
	that	the fridge_{acc}	Peter_{nom}	to repair	to try	promises
	dass	den Kühlschrank	zu reparieren	Peter	zu versuchen	verspricht
	that	the fridge_{acc}	to repair	Peter_{nom}	to try	promises
*	dass	den Kühlschrank	zu versuchen	Peter	zu reparieren	verspricht
	that	the fridge_{acc}	to try	Peter_{nom}	to repair	promises

Unordered Vector Grammars with Dominance Links

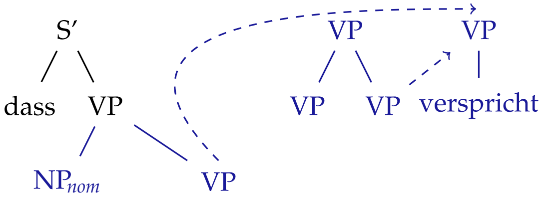
UVG-dls (Rambow, 1994a,b)



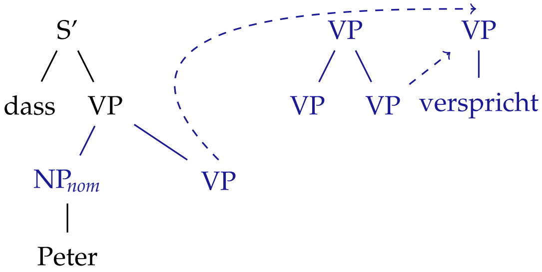
Example



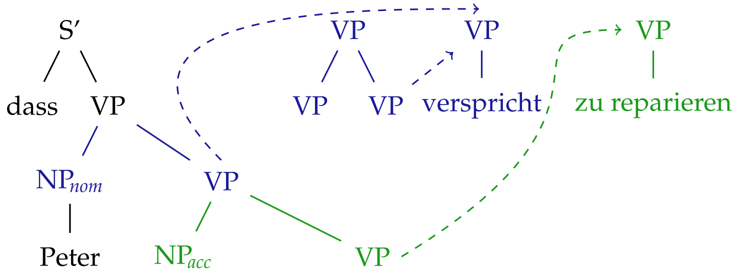
Example



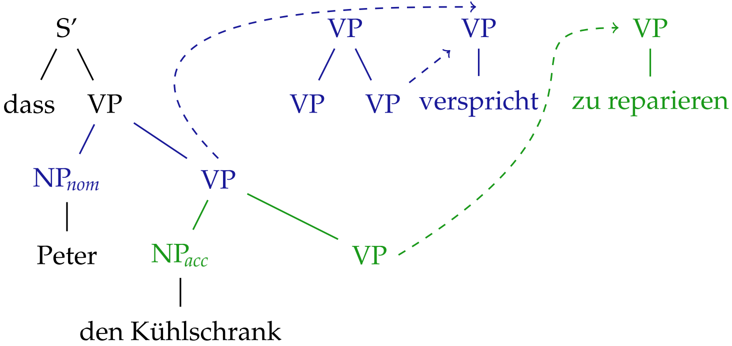
Example



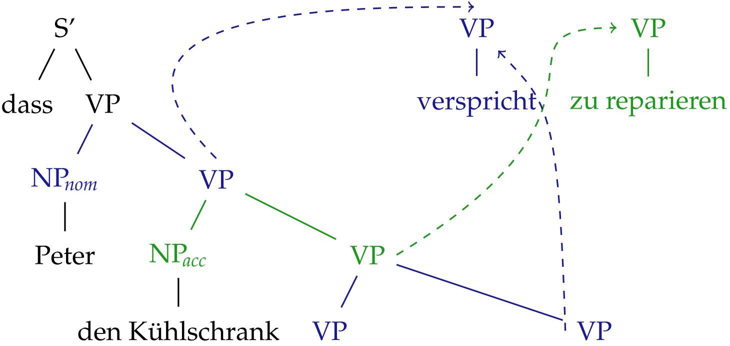
Example


S′
dass
VP
NP$_{nom}$
Peter
VP
NP$_{acc}$
VP
VP
VP
VP
VP
verspricht
VP
zu reparieren

Example

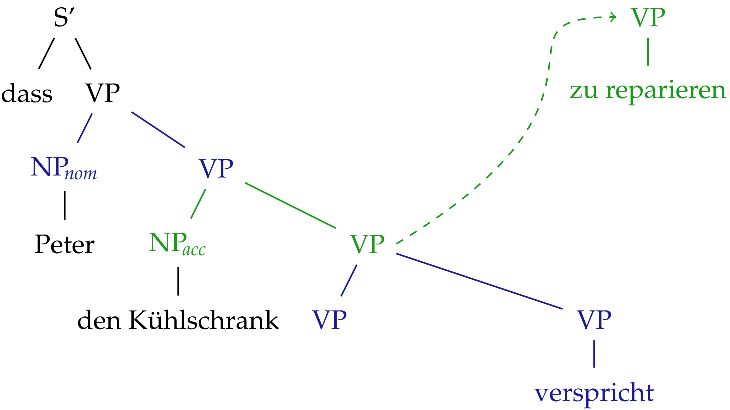

S′
dass
VP
NP_{nom}
Peter
VP
NP_{acc}
den Kühlschrank
VP
VP
VP
VP
VP
verspricht
VP
zu reparieren

Example

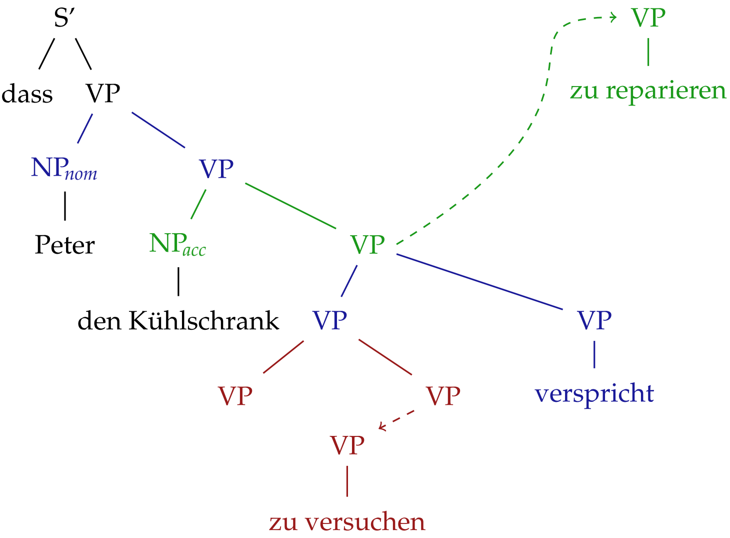


S′
dass
VP
NP$_{nom}$
Peter
VP
NP$_{acc}$
den Kühlschrank
VP
VP
VP
VP
verspricht
VP
zu reparieren

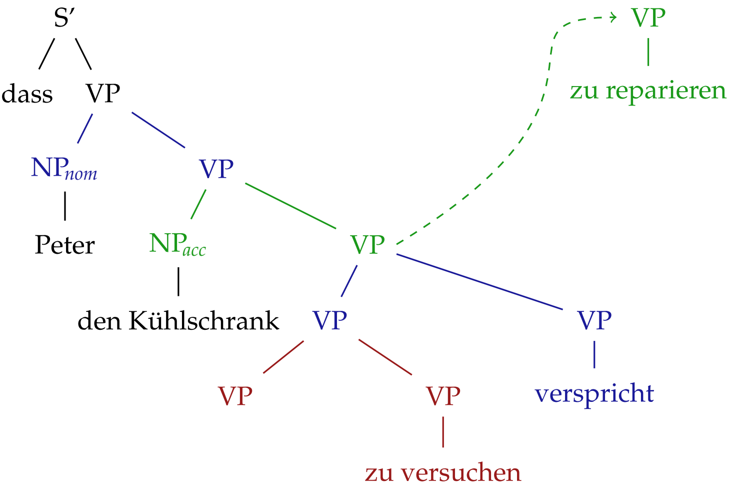
Example


S′
dass
VP
NP_nom
Peter
VP
NP_acc
den Kühlschrank
VP
VP
VP
verspricht
VP
zu reparieren

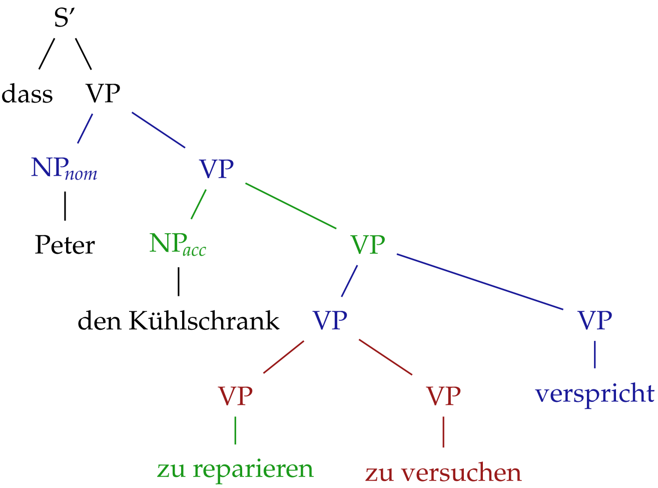
Example


S′
dass
VP
NP$_{nom}$
Peter
VP
NP$_{acc}$
den Kühlschrank
VP
VP
zu reparieren
VP
VP
verspricht
VP
VP
VP
zu versuchen

Example


S′
dass
VP
NP$_{nom}$
Peter
VP
NP$_{acc}$
den Kühlschrank
VP
VP
zu reparieren
VP
VP
VP
VP
zu versuchen
verspricht

Example


S′
dass
VP
NP$_{nom}$
Peter
VP
NP$_{acc}$
den Kühlschrank
VP
VP
VP
VP
VP
zu reparieren
zu versuchen
verspricht

Directly Related Formalisms

- D-tree substitution grammars (Rambow et al., 1995, 2001),
- graph-driven adjunction grammars (Candito and Kahane, 1998),
- tree description grammars (Kallmeyer, 2001),
- interaction grammars (Guillaume and Perrier, 2010),
- vector tree adjoining grammars (Becker et al., 1991; Rambow, 1994a),
- ...

Multiset-valued Linear Indexed Grammars

MLIGs (Rambow, 1994a,b)

Tuples $\mathcal{G} = \langle N, \Sigma, P, (S, \overline{x}_0) \rangle$:

- N: a finite set of *nonterminal* symbols,
- Σ: a finite *terminal* alphabet, disjoint from N,
- $V = (N \times \mathbb{N}^n) \uplus \Sigma$: the vocabulary,
- P: a finite set of *productions* in $(N \times \mathbb{N}^n) \times V^*$,
- $(S, \overline{x}_0) \in N \times \mathbb{N}^n$: the *axiom*.

Productions are more easily written as

$$(A, \overline{x}) \rightarrow u_0 (B_1, \overline{x}_1) u_1 \cdots u_m (B_m, \overline{x}_m) u_{m+1} \qquad (\star)$$

with each u_i in Σ^* and each $(B_i, \overline{x}_i)$ in $N \times \mathbb{N}^n$.

Multiset-valued Linear Indexed Grammars

MLIGs (Rambow, 1994a,b)

$$(A,\bar{x}) \to u_0(B_1,\bar{x}_1)u_1 \cdots u_m(B_m,\bar{x}_m)u_{m+1} \qquad (\star)$$

The *derivation* relation $\Rightarrow \subseteq V^* \times V^*$:

$$\delta(A,\bar{y})\delta' \Rightarrow \delta u_0(B_1,\bar{y}_1)u_1 \cdots u_m(B_m,\bar{y}_m)u_{m+1}\delta'$$

for some $\delta, \delta' \in V^*$, if

1. $\bar{x} \leqslant \bar{y}$,
2. $\forall 1 \leqslant i \leqslant m, \bar{x}_i \leqslant \bar{y}_i$,
3. $\bar{y} - \bar{x} = \sum_{i=1}^{m} \bar{y}_i - \bar{x}_i$.

Multiset-valued Linear Indexed Grammars

MLIGs (Rambow, 1994a,b)

The *language* of a MLIG is the set of terminal strings derived from $(S, \bar{x}_0)$, i.e.

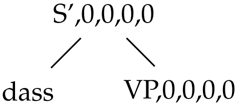
$$L(\mathcal{G}) = \{w \in \Sigma^* \mid (S, \bar{x}_0) \Rightarrow^* w\}$$

Example

A 4-dimensional MLIG with productions

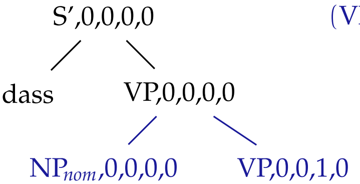
$$
\begin{aligned}
(S', 0, 0, 0, 0) &\to \text{dass } (\mathrm{VP}, 0, 0, 0, 0) \\
(\mathrm{NP}_{nom}, 0, 0, 0, 0) &\to \text{Peter} \\
(\mathrm{NP}_{acc}, 0, 0, 0, 0) &\to \text{den Kühlschrank} \\
(\mathrm{VP}, 0, 0, 0, 0) &\to (\mathrm{VP}, 0, 0, 0, 0)\ (\mathrm{VP}, 1, 0, 0, 0) \\
(\mathrm{VP}, -1, 0, 0, 0) &\to \text{zu versuchen} \\
(\mathrm{VP}, 0, 0, 0, 0) &\to (\mathrm{NP}_{acc}, 0, 0, 0, 0)\ (\mathrm{VP}, 0, 1, 0, 0) \\
(\mathrm{VP}, 0, -1, 0, 0) &\to \text{zu reparieren} \\
(\mathrm{VP}, 0, 0, 0, 0) &\to (\mathrm{NP}_{nom}, 0, 0, 0, 0)\ (\mathrm{VP}, 0, 0, 1, 0) \\
(\mathrm{VP}, 0, 0, 0, 0) &\to (\mathrm{VP}, 0, 0, 0, 0)\ (\mathrm{VP}, 0, 0, 0, 1) \\
(\mathrm{VP}, 0, 0, -1, -1) &\to \text{verspricht}
\end{aligned}
$$

Example



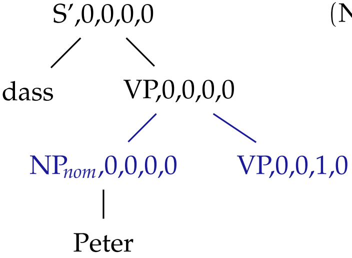
$$(S', 0, 0, 0, 0) \rightarrow \text{dass}\ (VP, 0, 0, 0, 0)$$

Example



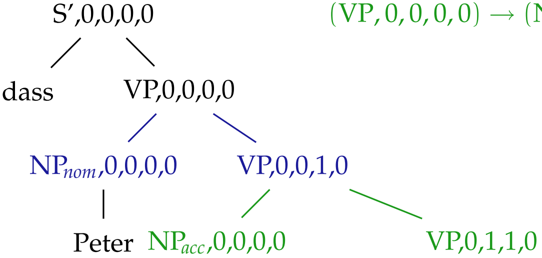
$$(\mathrm{VP}, 0, 0, 0, 0) \rightarrow (\mathrm{NP}_{nom}, 0, 0, 0, 0)\ (\mathrm{VP}, 0, 0, 1, 0)$$

Example



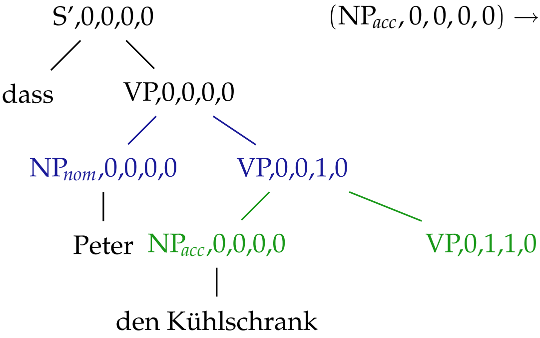
$$(NP_{nom}, 0, 0, 0, 0) \rightarrow \text{Peter}$$

Example



$$(\mathrm{VP}, 0, 0, 0, 0) \rightarrow (\mathrm{NP}_{acc}, 0, 0, 0, 0)\ (\mathrm{VP}, 0, 1, 0, 0)$$

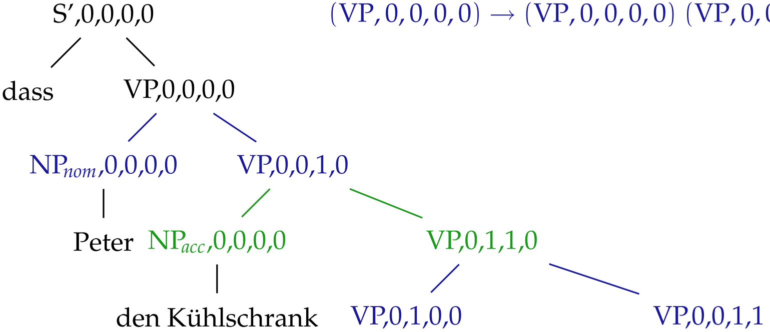
Example



$(\text{NP}_{acc}, 0, 0, 0, 0) \rightarrow$ den Kühlschrank

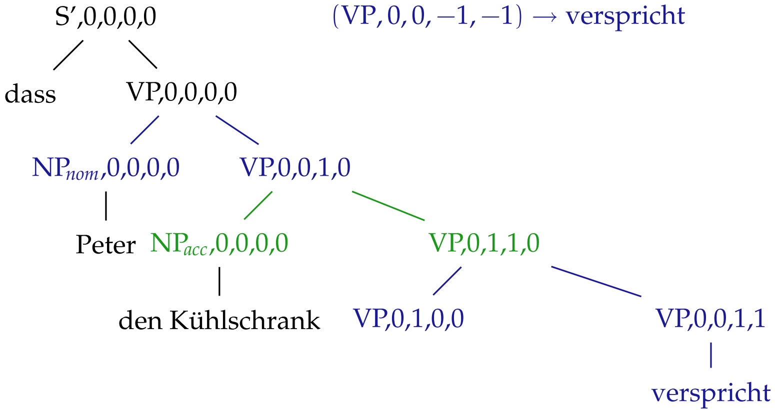
Example

$$(\mathrm{VP}, 0, 0, 0, 0) \rightarrow (\mathrm{VP}, 0, 0, 0, 0)\ (\mathrm{VP}, 0, 0, 0, 1)$$



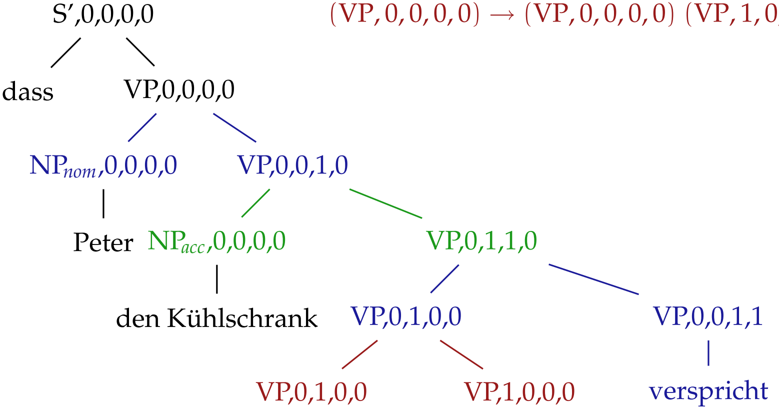
Example

$$(\mathrm{VP}, 0, 0, -1, -1) \rightarrow \text{verspricht}$$



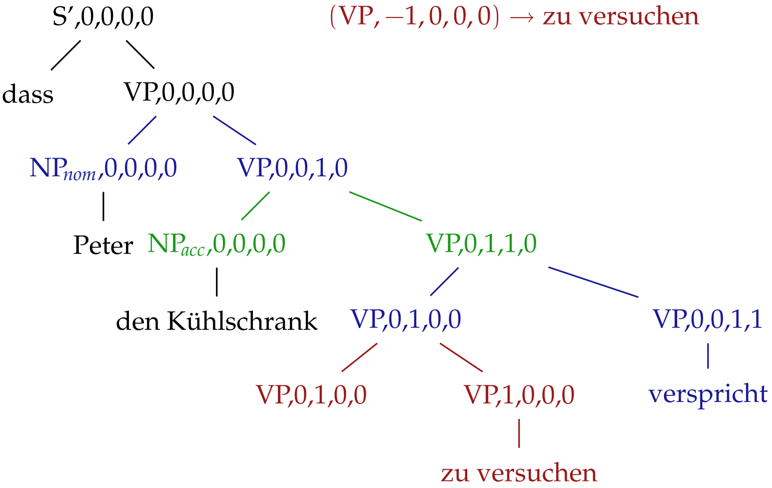
Example

$$(\mathrm{VP}, 0, 0, 0, 0) \rightarrow (\mathrm{VP}, 0, 0, 0, 0)\ (\mathrm{VP}, 1, 0, 0, 0)$$



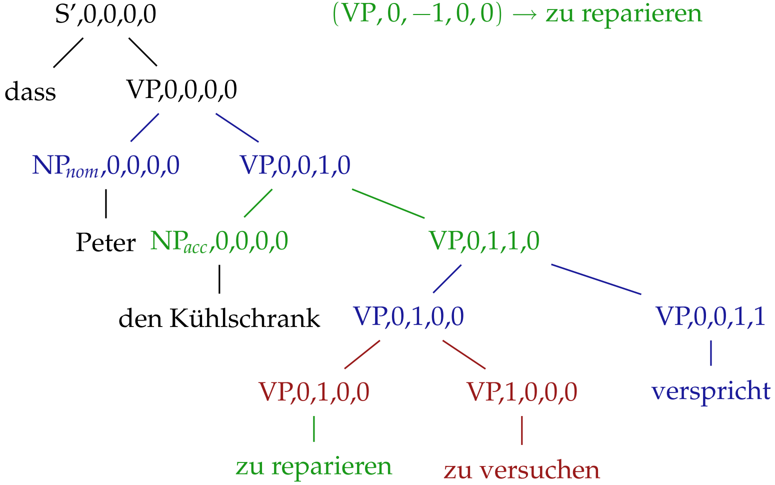
Example

$(\mathrm{VP}, -1, 0, 0, 0) \rightarrow$ zu versuchen



Example

$$(VP, 0, -1, 0, 0) \rightarrow \text{zu reparieren}$$



Equivalence with UVG-dls

Theorem (Rambow, 1994b)

Every MLIG can be transformed into an equivalent UVG-dl in logarithmic space, and conversely.

Proof idea: from UVG-dls.

1. Convert the UVG-dl to *strict* form, with vectors of productions connected by the dominance links,
2. each coordinate in $\mathbb{N}^n$ encodes a dominance link. □

Equivalence with UVG-dls

Proof idea: to UVG-dls.

1. Convert the MLIG to *ordinary form*: all coordinates in vectors hold values in $\{0,1\}$,
2. convert now to *restricted index normal form*:
   - $(A,\overline{0}) \to \alpha$, $\alpha \in (\Sigma \cup (N \times \{\overline{0}\}))^*$,
   - $(A,\overline{0}) \to (B,\overline{e}_i)$, or
   - $(A,\overline{e}_i) \to (B,\overline{0})$;
3. - pair in UVG-dl vectors productions of forms $(A,\overline{0}) \to (B,\overline{e}_i)$ and $(C,\overline{e}_i) \to (D,\overline{0})$;
   - productions of form $(A,\overline{0}) \to \alpha$ result in singleton vectors. $\square$

Related Formalisms (1)

MLIGs are *exactly* the same as

- vector addition tree automata (de Groote et al., 2004), and
- branching vector addition systems with states (Verma and Goubault-Larrecq, 2005).
- *They are indeed a natural generalization of Petri nets/vector addition systems.*

Related Formalisms (1)

MLIGs are *exactly* the same as

- vector addition tree automata (de Groote et al., 2004), and
- branching vector addition systems with states (Verma and Goubault-Larrecq, 2005).
- *They are indeed a natural generalization of Petri nets/vector addition systems.*

Related Formalisms (2)

MLIG emptiness and membership reduce to

- emptiness and membership in UVG-dls (and related formalisms),
- provability in multiplicative exponential linear logic (de Groote et al., 2004),
- emptiness and membership of abstract categorial grammars (de Groote et al., 2004; Yoshinaka and Kanazawa, 2005),

Related Formalisms (3)

MLIG emptiness and membership reduce to

- emptiness and membership of Stabler (1997)'s minimalist grammars without shortest move constraint (Salvati, 2010),
- satisfiability of first-order logic on data trees (Bojańczyk et al., 2009).
- *Their decidability is a central open problem.*

Related Formalisms (3)

MLIG emptiness and membership reduce to

- emptiness and membership of Stabler (1997)'s minimalist grammars without shortest move constraint (Salvati, 2010),
- satisfiability of first-order logic on data trees (Bojańczyk et al., 2009).
- *Their decidability is a central open problem.*

Summary

Problem	Lower bound	Upper bound
MLIG Emptiness, MLIG Membership	2ExpSpace (Lazić, 2010)	Not known to be decidable
kb-MLIG Emptiness, kb-MLIG Membership	ExpTime (this talk)	ExpTime (this talk)
{MLIG_ℓ, kb-MLIG_ℓ} Membership	NPTime (Koller and Rambow, 2007)	NPTime (trivial)
kr-MLIG {Emptiness, Membership}	PTime (Jones and Laaser, 1976)	PTime (this talk)
MLIG Boundedness	2ExpTime (Demri et al., 2009)	2ExpTime (Demri et al., 2009)
MLIG k-Boundedness	ExpTime (this talk)	ExpTime (this talk)

k-Ranked MLIGs

A MLIG derivation

$$\alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_p$$

is of *rank* k for some $k \geqslant 0$ if no vector with a *sum* of components larger than k can appear in any α_j.

A MLIG is k-*ranked* (noted kr-MLIG) if any derivation starting with $\alpha_0 = (S, \overline{x}_0)$ is of rank k.

k-Ranked MLIGs

Lemma

Any n-dimensional k-ranked MLIG $\mathcal{G}$ can be transformed into an equivalent CFG $\mathcal{G}'$ in time $O(|\mathcal{G}| \cdot (n+1)^{k^3})$.

Proof idea:

1. Convert to *extended two form*:
   terminal $(A, \overline{0}) \to a$, $(A, \overline{0}) \to \varepsilon$,
   nonterminal $(A, \overline{x}) \to (B_1, \overline{x}_1)(B_2, \overline{x}_2)$,
   $(A, \overline{x}) \to (B_1, \overline{x}_1)$,
2. at most $|N| \cdot (n+1)^k$ nonterminals $(A, \overline{y})$ in $N' \subseteq N \times \mathbb{N}^n$ with $\sum_{i=1}^{n} \overline{y}(i) \leqslant k$,
3. at most $(n+1)^{k^3}$ choices of nonterminals in N' for a production $(A, \overline{x}) \to (B_1, \overline{x}_1)(B_2, \overline{x}_2)$. $\square$

Summary

Problem	Lower bound	Upper bound
MLIG Emptiness, MLIG Membership	2ExpSpace (Lazić, 2010)	Not known to be decidable
kb-MLIG Emptiness, kb-MLIG Membership	ExpTime (this talk)	ExpTime (this talk)
{MLIG_ℓ, kb-MLIG_ℓ} Membership	NPTime (Koller and Rambow, 2007)	NPTime (trivial)
kr-MLIG {Emptiness, Membership}	PTime (Jones and Laaser, 1976)	PTime (this talk)
MLIG Boundedness	2ExpTime (Demri et al., 2009)	2ExpTime (Demri et al., 2009)
MLIG k-Boundedness	ExpTime (this talk)	ExpTime (this talk)

k-Bounded MLIGs

A MLIG derivation

$$\alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_p$$

is k-*bounded* for some $k \geqslant 0$ if no vector with a *coordinate* larger than k can appear in any α_j.

A MLIG is k-*bounded* (noted kb-MLIG) if any derivation starting with $\alpha_0 = (S, \overline{x}_0)$ is k-bounded.

k-Bounded MLIGs

Lemma

Any n-dimensional k-bounded MLIG $\mathcal{G}$ can be transformed into an equivalent CFG $\mathcal{G}'$ in time $O(|\mathcal{G}| \cdot (k+1)^{n^2})$.

Proof idea:

1. Convert to *extended two form*,
2. $N' \subseteq N \times \{0, \ldots, k\}^n$, $|N'| \leqslant |N| \cdot (k+1)^n$,
3. - $(A, \overline{x}) \to (B_1, \overline{x}_1)(B_2, \overline{x}_2)$,
   - $(A, \overline{y}) \in N'$,
   - $0 < i \leqslant n$,

   result in $\leqslant k+1$ ways to split $(\overline{y}(i) - \overline{x}(i)) \leqslant k$ into $\overline{y}_1(i) + \overline{y}_2(i)$ in production $(A, \overline{y}) \to (B_1, \overline{x}_1 + \overline{y}_1)(B_2, \overline{x}_2 + \overline{y}_2)$. $\square$

k-Bounded MLIGs

Theorem

Emptiness and membership for k*-bounded MLIGs are* ExpTime*-complete, even for fixed* $k \geqslant 1$*.*

- lower bound: by encoding computations of alternating Turing machines running in polynomial space,
- upper bound: by the previous lemma.

Summary

Problem	Lower bound	Upper bound
MLIG Emptiness, MLIG Membership	2ExpSpace (Lazić, 2010)	Not known to be decidable
kb-MLIG Emptiness, kb-MLIG Membership	ExpTime (this talk)	ExpTime (this talk)
{MLIG$_\ell$, kb-MLIG$_\ell$} Membership	NPTime (Koller and Rambow, 2007)	NPTime (trivial)
kr-MLIG {Emptiness, Membership}	PTime (Jones and Laaser, 1976)	PTime (this talk)
MLIG Boundedness	2ExpTime (Demri et al., 2009)	2ExpTime (Demri et al., 2009)
MLIG k-Boundedness	ExpTime (this talk)	ExpTime (this talk)

Boundedness

A MLIG is

- *bounded* if $\exists$k s.t. it is k-bounded,
- *ranked* if $\exists$k s.t. it is k-ranked:

$$\text{bounded} \quad \Leftrightarrow \quad \text{ranked}$$

Theorem (Demri et al., 2009)

Boundedness for MLIGs is 2ExpTime*-complete.*

Summary

Problem	Lower bound	Upper bound
MLIG Emptiness, MLIG Membership	2ExpSpace (Lazić, 2010)	Not known to be decidable
kb-MLIG Emptiness, kb-MLIG Membership	ExpTime (this talk)	ExpTime (this talk)
{MLIG$_\ell$, kb-MLIG$_\ell$} Membership	NPTime (Koller and Rambow, 2007)	NPTime (trivial)
kr-MLIG {Emptiness, Membership}	PTime (Jones and Laaser, 1976)	PTime (this talk)
MLIG Boundedness	2ExpTime (Demri et al., 2009)	2ExpTime (Demri et al., 2009)
MLIG k-Boundedness	ExpTime (this talk)	ExpTime (this talk)

k-Boundedness

Corollary

Let $k \geqslant 1$*;* k*-boundedness for MLIGs is* ExpTime*-complete.*

- lower bound: by the ExpTime-hardness of emptiness and membership in 1-bounded MLIGS,
- upper bound: by converting into a CFG.

Summary

Problem	Lower bound	Upper bound
MLIG Emptiness, MLIG Membership	2EXPSPACE (Lazić, 2010)	Not known to be decidable
kb-MLIG Emptiness, kb-MLIG Membership	EXPTIME (this talk)	EXPTIME (this talk)
$\{\text{MLIG}_\ell, \text{kb-MLIG}_\ell\}$ Membership	NPTIME (Koller and Rambow, 2007)	NPTIME (trivial)
kr-MLIG {Emptiness, Membership}	PTIME (Jones and Laaser, 1976)	PTIME (this talk)
MLIG Boundedness	2EXPTIME (Demri et al., 2009)	2EXPTIME (Demri et al., 2009)
MLIG k-Boundedness	EXPTIME (this talk)	EXPTIME (this talk)

Lexicalized MLIGs

A terminal MLIG derivation

$$\alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_p = w$$

is c-*lexicalized* for some $c \geqslant 0$ if $p \leqslant c \cdot |w|$.

A MLIG is *lexicalized* if there exists c such that any terminal derivation starting from $(S, \bar{x}_0)$ is c-lexicalized.

Note: This captures lexicalization in UVG-dls.

Lexicalized MLIGs

Theorem (Koller and Rambow, 2007)

Uniform membership of $\langle \mathcal{G}, w \rangle$ for $\mathcal{G}$ a 1-bounded, lexicalized, UVG-dl with finite language is NPTIME*-hard, even for* $|w| = 1$.

By a reduction from the *normal dominance graph configurability* problem (Althaus et al., 2003).

Summary

Problem	Lower bound	Upper bound
{MLIG, MLIG_ℓ} Emptiness, MLIG Membership	2ExpSpace (Lazić, 2010)	Not known to be decidable
{kb-MLIG, kb-MLIG_ℓ} Emptiness, kb-MLIG Membership	ExpTime (this talk)	ExpTime (this talk)
{MLIG_ℓ, kb-MLIG_ℓ} Membership	NPTime (Koller and Rambow, 2007)	NPTime (trivial)
kr-MLIG {Emptiness, Membership}	PTime (Jones and Laaser, 1976)	PTime (this talk)
{MLIG, MLIG_ℓ} Boundedness	2ExpTime (Demri et al., 2009)	2ExpTime (Demri et al., 2009)
{MLIG, MLIG_ℓ} k-Boundedness	ExpTime (this talk)	ExpTime (this talk)

Concluding Remarks

About *mild context-sensitivity*:

1. support for *limited cross-serial dependencies*: seems doubtful, Rambow (1994a) conjectures $L_{copy} = \{ww \mid w \in \{a, b\}^*\}$ is not derivable,
2. *semilinearity*: does not hold, from the Petri net literature,
3. *polynomial recognition*: only for k-ranked grammars.

But all 3 hold e.g. for k-ranked V-TAG.

References

Althaus, E., Duchier, D., Koller, A., Mehlhorn, K., Niehren, J., and Thiel, S., 2003. An efficient graph algorithm for dominance constraints. *Journal of Algorithms*, 48(1):194–219. doi:10.1016/S0196-6774(03)00050-6.

Becker, T., Joshi, A.K., and Rambow, O., 1991. Long-distance scrambling and tree adjoining grammars. In *EACL'91*, pages 21–26. ACL Press. doi:10.3115/977180.977185.

Bojańczyk, M., Muscholl, A., Schwentick, T., and Segoufin, L., 2009. Two-variable logic on data trees and XML reasoning. *Journal of the ACM*, 56(3):1–48. doi:10.1145/1516512.1516515.

Candito, M.H. and Kahane, S., 1998. Defining DTG derivations to get semantic graphs. In *TAG+4*, pages 25–28.

de Groote, P., Guillaume, B., and Salvati, S., 2004. Vector addition tree automata. In *LICS 2004*, pages 64–73. IEEE Computer Society. ISBN 0-7695-2192-4. doi:10.1109/LICS.2004.51.

Demri, S., Jurdziński, M., Lachish, O., and Lazić, R., 2009. The covering and boundedness problems for branching vector addition systems. In Kannan, R. and Narayan Kumar, K., editors, *FSTTCS 2009*, volume 4 of *LIPIcs*, pages 181–192. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSTTCS.2009.2317.

Guillaume, B. and Perrier, G., 2010. Interaction grammars. *Research on Language and Computation*. doi:10.1007/s11168-010-9066-x. To appear.

Jones, N.D. and Laaser, W.T., 1976. Complete problems for deterministic polynomial time. *Theor. Comput. Sci.*, 3(1):105–117. doi:10.1016/0304-3975(76)90068-2.

Kallmeyer, L., 2001. Local tree description grammars. *Grammars*, 4(2): 85–137. doi:10.1023/A:1011431526022.

Koller, A. and Rambow, O., 2007. Relating dominance formalisms. In *FG 2007*.

Lazić, R., 2010. The reachability problem for branching vector addition systems requires doubly-exponential space. *Information Processing Letters*. doi:10.1016/j.ipl.2010.06.008. To appear.

Lichte, T., 2007. An MCTAG with tuples for coherent constructions in German. In *FG 2007*.

Rambow, O., 1994a. *Formal and Computational Aspects of Natural Language Syntax*. PhD thesis, University of Pennsylvania.

Rambow, O., 1994b. Multiset-valued linear index grammars: imposing dominance constraints on derivations. In *ACL'94*, pages 263–270. ACL Press. doi:10.3115/981732.981768.

Rambow, O., Vijay-Shanker, K., and Weir, D., 1995. D-tree grammars. In *ACL'95*, pages 151–158. ACL Press. doi:10.3115/981658.981679.

Rambow, O., Weir, D., and Vijay-Shanker, K., 2001. D-tree substitution grammars. *Comput. Linguist.*, 27(1):89–121. doi:10.1162/089120101300346813.

Salvati, S., 2010. Minimalist grammars in the light of logic. Manuscript.

Stabler, E.P., 1997. Derivational minimalism. In Retoré, C., editor, *LACL'96*, volume 1328 of *LNCS*, pages 68–95. Springer. doi:10.1007/BFb0052147.

Verma, K.N. and Goubault-Larrecq, J., 2005. Karp-Miller trees for a branching extension of VASS. *Discrete Mathematics and Theoretical Computer Science*, 7(1):217–230. http://www.dmtcs.org/volumes/abstracts/dm070113.abs.html.

Yoshinaka, R. and Kanazawa, M., 2005. The complexity and generative capacity of lexicalized abstract categorial grammars. In Blache, P., Stabler, E., Busquets, J., and Moot, R., editors, *LACL 2005*, volume 3492 of *LNCS*, pages 330–346. Springer. doi:10.1007/11422532_22.