Linguistique Computationnelle: Analyse et Génération d'Enoncés à l'Aide de Diagrammes de Van Kampen

Marc Dymetman

Xerox Research Centre Europe

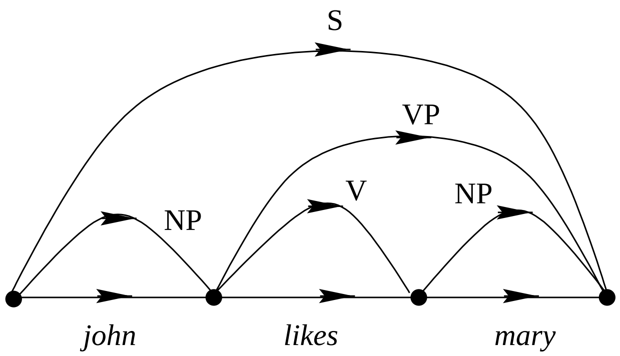
Grenoble, France

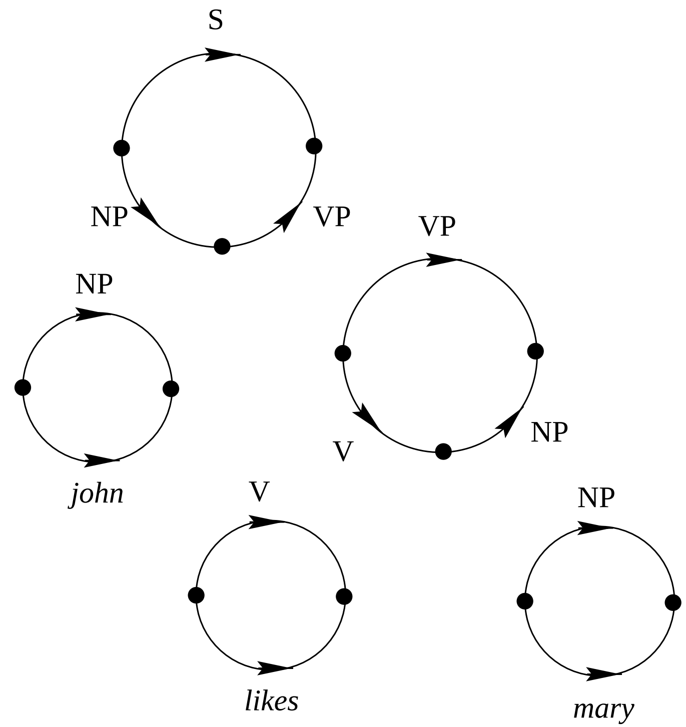
Introduction

- Origin of this work: aspects reminiscent of group theory in Categorial Grammars (Ajdukiewicz, Lambek) and in Linear Logic
- Is it possible to base a theory of linguistic description solely on group structure? YES!
- Interplay between algebra (Group Theory) and geometry (Van Kampen Diagrams)
- Diagrams as a generalization of charts (context-free, DCGs Colmerauer's Q-systems)
  - Symmetrical account of grammar: parsing / generation
  - Long-distance movement in grammar
- Vista on underlying connections between group theory, "graphical computation", and cognitive processes...

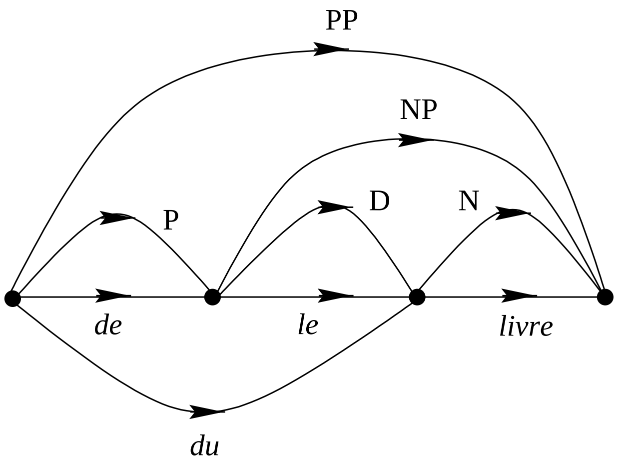
The Geometric Viewpoint

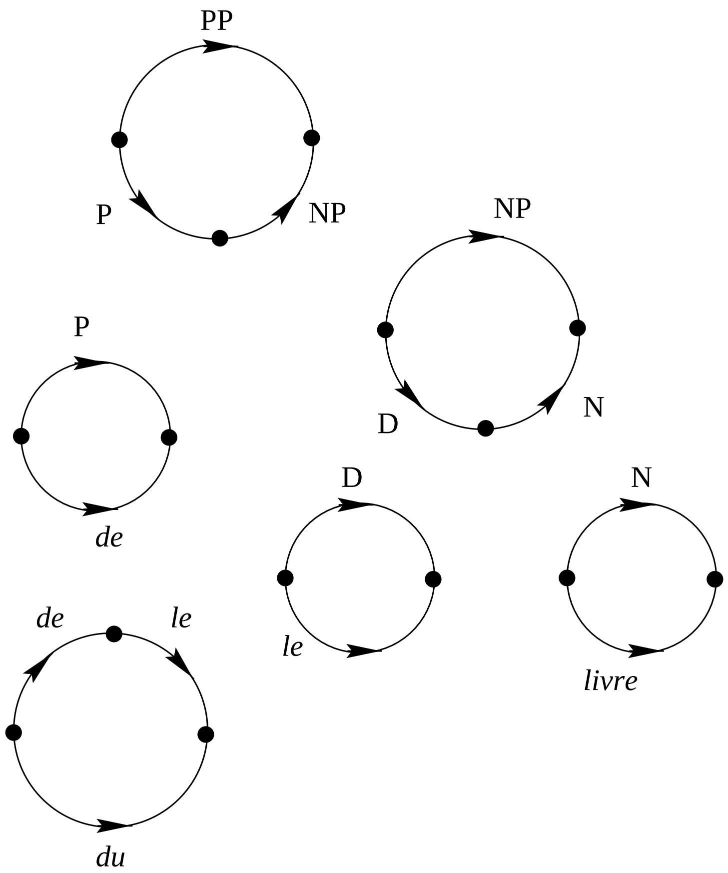
Charts and cells



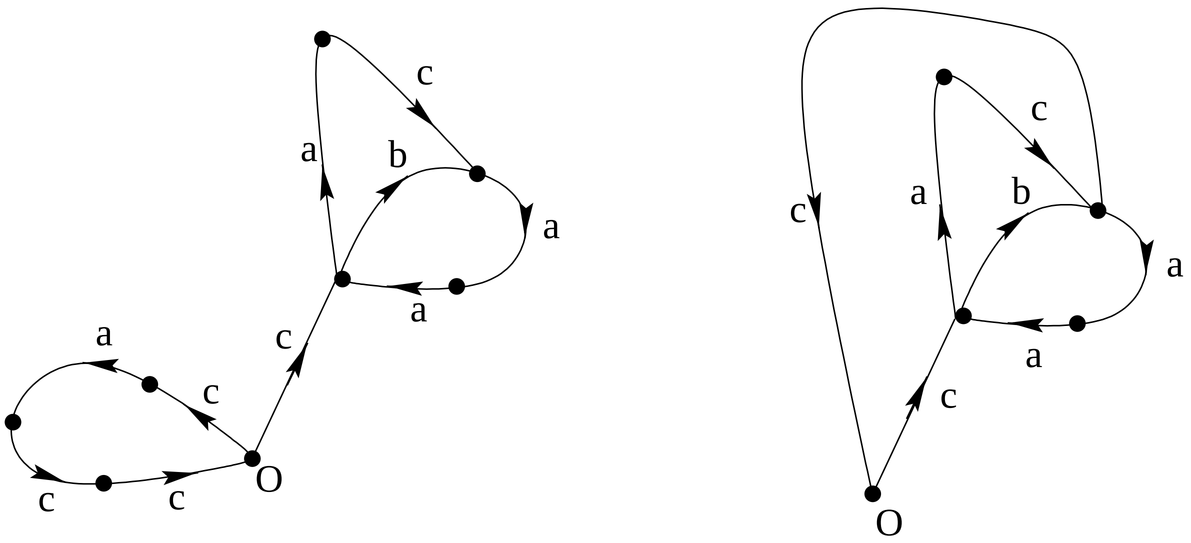


Charts and cells (cont.)





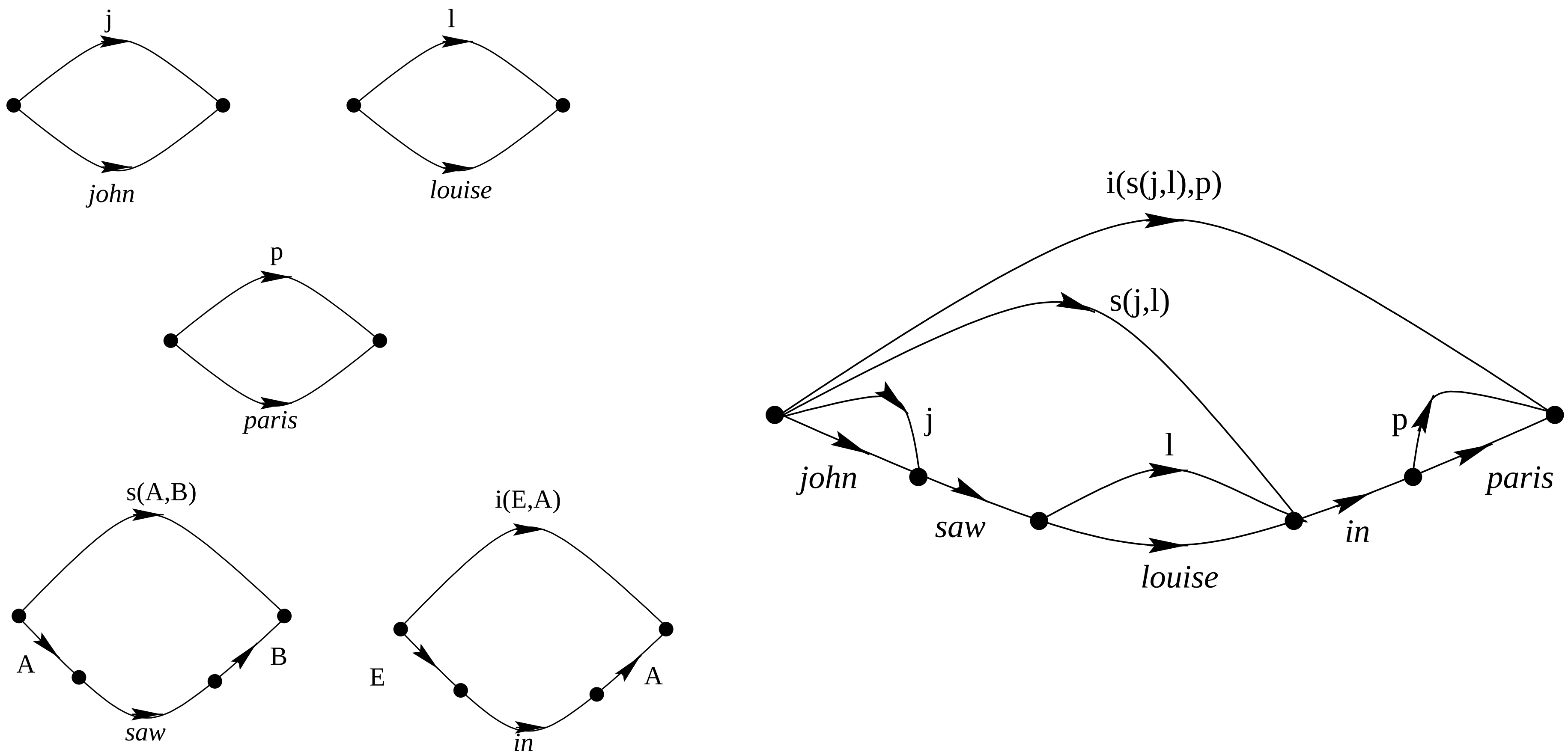
Cancellation Diagrams



Diagrams as computational devices

- "Program" : a fixed collection of cells C over a vocabulary V;
- "Computation" : a reduced diagram D over C;
- "Result" : a boundary word w of a diagram in D; w is an element of the free group $F(V)$ generated by V.

Grammars as Group Computation Structures: geometric view



The Algebraic Viewpoint

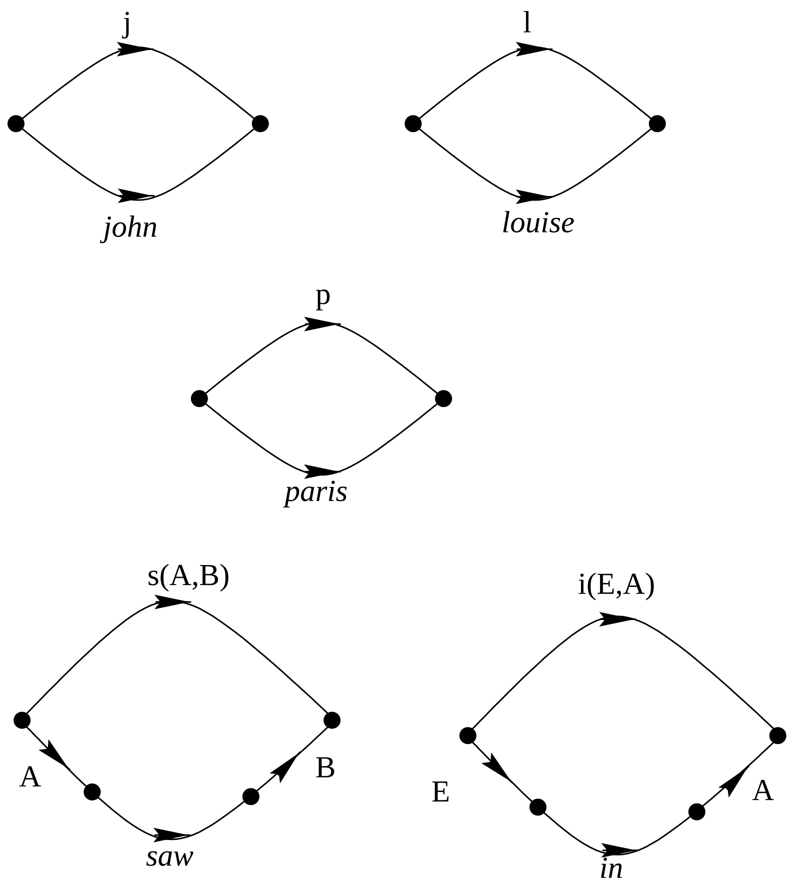
Some algebraic notions

free group $a\ b\ a^{-1}\ c\ c^{-1}\ b\ c^{-1}\ c\ c^{-1}\ a \ = \ a\ b\ a^{-1}\ b\ c^{-1}\ a$

conjugacy x, x' are *conjugate* in G if there exists $y \in G$ such that $x' = yxy^{-1}$.

normality A *normal* subset S of G is a set which contains along with an element all its conjugates. Special cases: *normal subgroup*, *normal submonoid.*

Grammars as Group Computation Structures: algebraic view



`j`	$john^{-1}$
`l`	$louise^{-1}$
`p`	$paris^{-1}$
`s(A,B)`	$\mathtt{B}^{-1} saw^{-1} \mathtt{A}^{-1}$
`i(E,A)`	$\mathtt{A}^{-1} in^{-1} \mathtt{E}^{-1}$

Grammars as Group Computation Structures: algebraic view (cont.)

$$\mathtt{j}\ \mathit{john}^{-1}$$
$$\mathtt{l}\ \mathit{louise}^{-1}$$
$$\mathtt{p}\ \mathit{paris}^{-1}$$
$$\mathtt{s(A,B)}\ \mathtt{B}^{-1}\mathit{saw}^{-1}\mathtt{A}^{-1}$$
$$\mathtt{i(E,A)}\ \mathtt{A}^{-1}\mathit{in}^{-1}\mathtt{E}^{-1}$$

$$\mathtt{r}_1 = \mathtt{i(s(j,l),p)}\ \mathtt{p}^{-1}\ \mathit{in}^{-1}\ \mathtt{s(j,l)}^{-1}$$
$$\mathtt{r}_2 = \mathtt{s(j,l)}\ \mathtt{l}^{-1}\ \mathit{saw}^{-1}\ \mathtt{j}^{-1}$$
$$\mathtt{r}_3 = \mathtt{j}\ \mathit{john}^{-1}$$
$$\mathtt{r}_4 = \mathtt{l}\ \mathit{louise}^{-1}$$
$$\mathtt{r}_5 = \mathtt{p}\ \mathit{paris}^{-1}$$

$$\mathtt{r}_1 \cdot \mathtt{r}_2 \cdot \mathtt{r}_3 \cdot (\mathit{john\ saw})\ \mathtt{r}_4\ (\mathit{john\ saw})^{-1} \cdot (\mathit{john\ saw\ louise\ in})\ \mathtt{r}_5\ (\mathit{john\ saw\ louise\ in})^{-1}$$
$$=$$
$$\mathtt{i(s(j,l),p)}\ \mathit{paris}^{-1}\ \mathit{in}^{-1}\ \mathit{louise}^{-1}\ \mathit{saw}^{-1}\ \mathit{john}^{-1}$$

Algebraic view of computation: Group Computation Structure

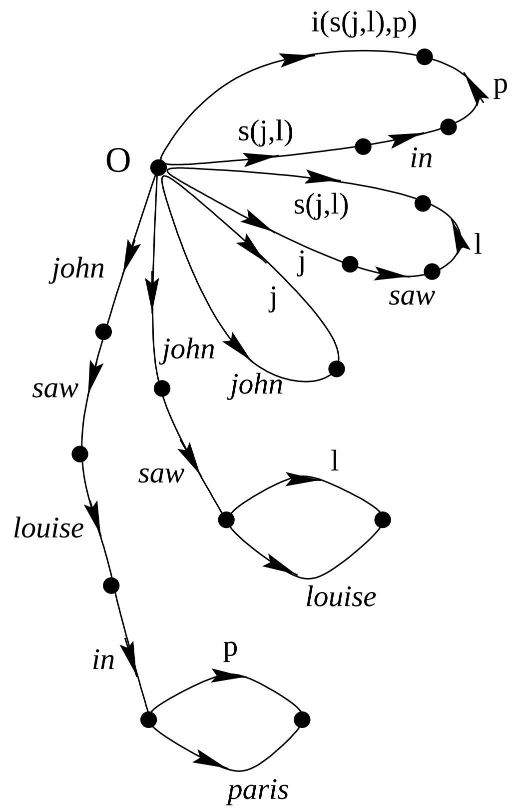
- "Program" : a fixed collection R of "relators" r_i over a vocabulary V, where each r_i is an element of $F(V)$;
- "Computation" : a product of conjugates of relators, that is, a product of the form $x_1 r_1 x_1{}^{-1} \cdots x_n r_n x_n{}^{-1}$ where the r_i are relators, and the x_i are arbitrary elements of $F(V)$;
- "Result" : the reduced word in $F(V)$ which is equal to the value of a computation.
  - The set of results is equal to the smallest normal submonoid of $F(V)$ containing R (*normal submonoid closure* of R in $F(V)$).
  - Special case: if R is symmetrical, that is, $r \in R \Rightarrow r^{-1} \in R$, then the set of results is the *normal subgroup closure* of R in $F(V)$.

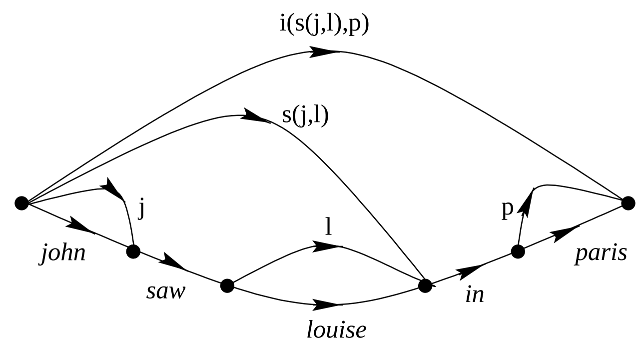
Equivalence between the geometric and the algebraic views

Fundamental theorem of combinatorial group theory (Van Kampen 1933, Lyndon & Schupp 1977):

> *If C is a collection of cells c_i, R the collection of the boundaries r_i of the c_i's, then w is a result of C in the geometrical sense iff it is a result of R in the algebraic sense.*

Equivalence between the geometric and the algebraic views: the role of conjugates





An example of a commutative Group Computation Structure: Logic Programs

Prolog:

```
path(X,Z) :- arc(X,Y), path(Y,Z).
path(X,X).
arc(a,b).
arc(b,c).
```

GCS:

$$\texttt{path(X,Z)}\ \texttt{path(Y,Z)}^{-1}\ \texttt{arc(X,Y)}^{-1}$$

```
path(X,X)
arc(a,b)
arc(b,c)
```

$$\texttt{path(a,c)} = \texttt{path(a,c)}\ \ \texttt{path(b,c)}^{-1}\ \ \texttt{arc(a,b)}^{-1}$$

$$\texttt{arc(a,b)}$$

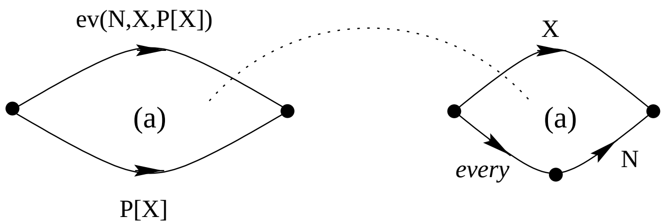
$$\texttt{path(b,c)}\ \ \texttt{path(c,c)}^{-1}\ \ \texttt{arc(b,c)}^{-1}$$

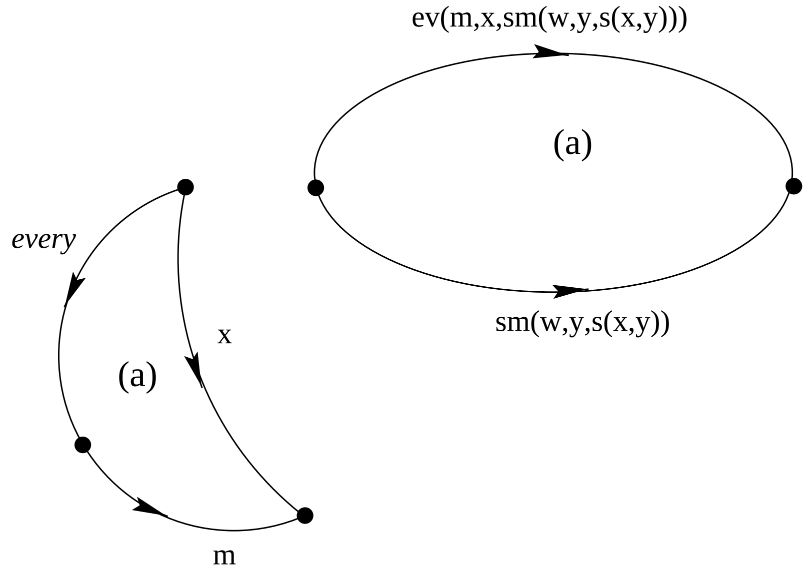
$$\texttt{arc(b,c)}$$

$$\texttt{path(c,c)}$$

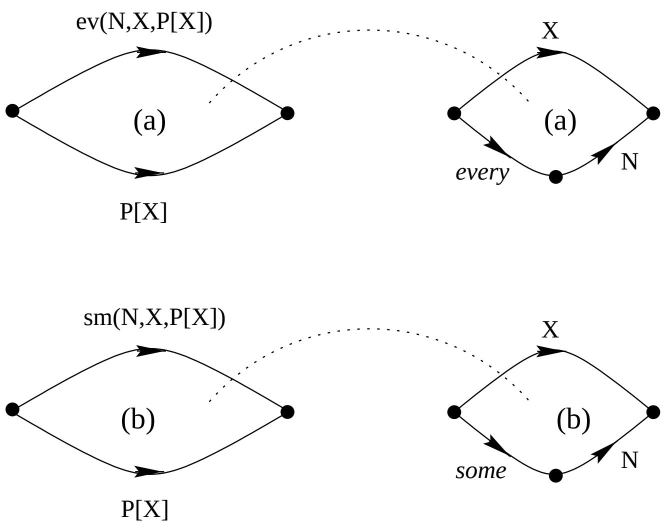
Diagrams and Non-Bounded Phenomena in Grammar

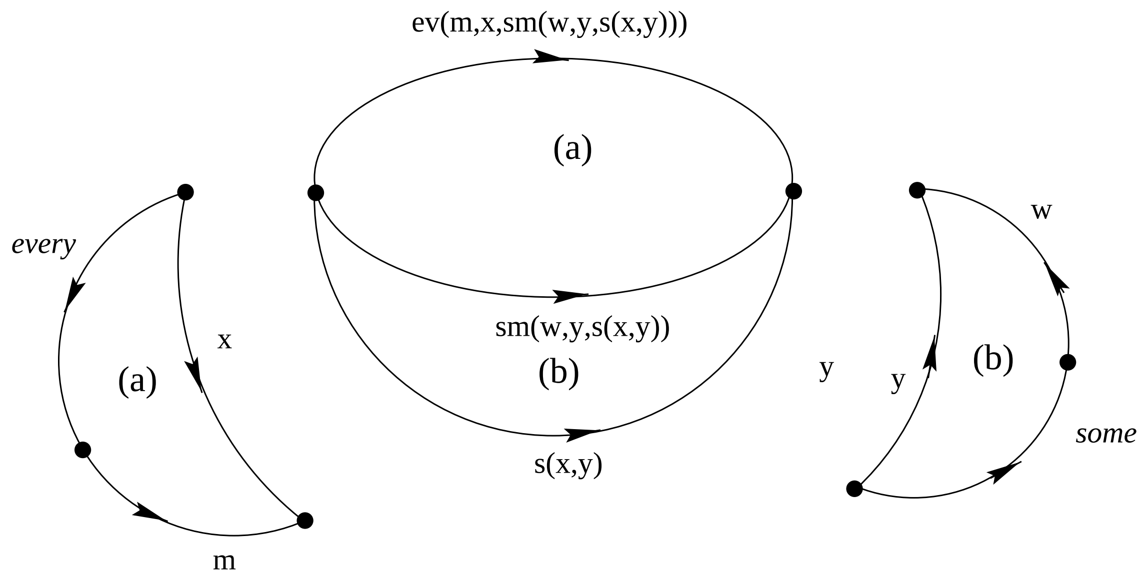
Multi-relators and non-boundedness: quantifier scoping



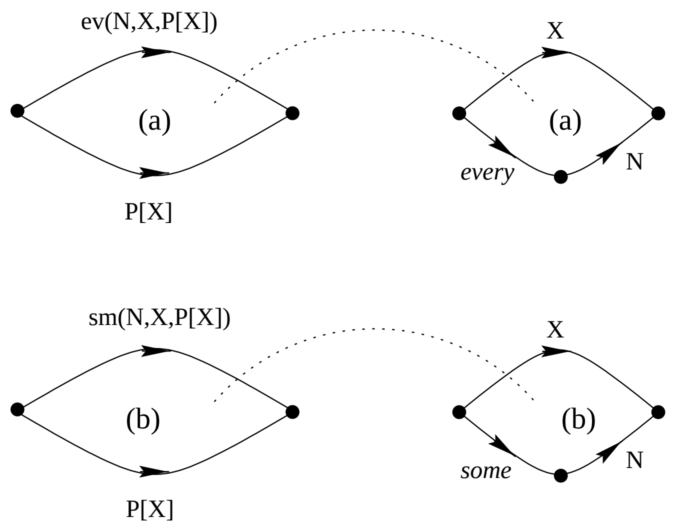


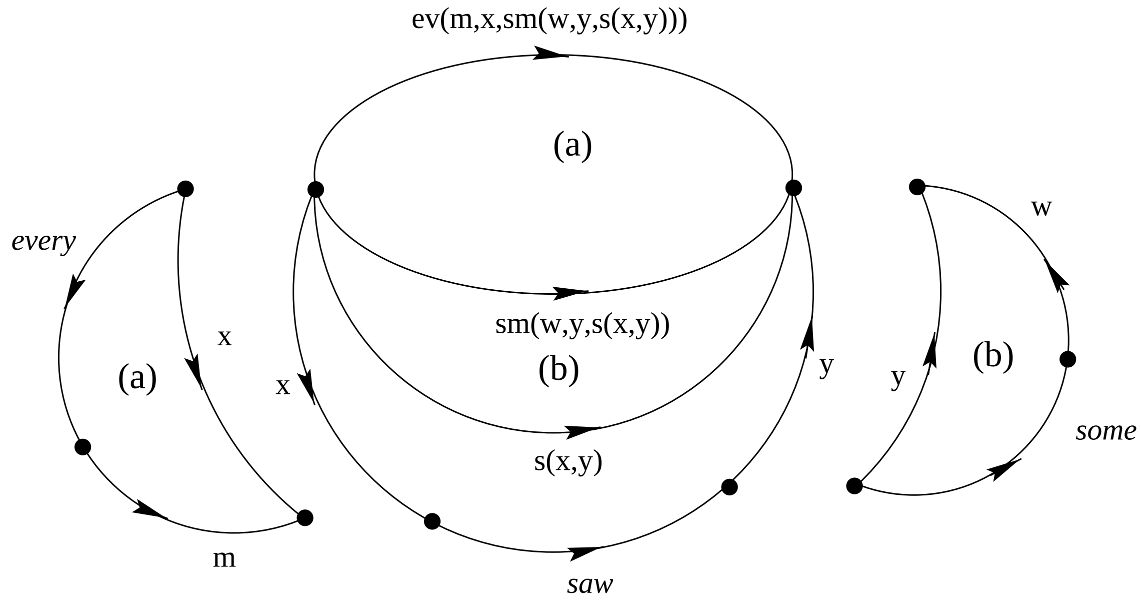
Multi-relators and non-boundedness: quantifier scoping



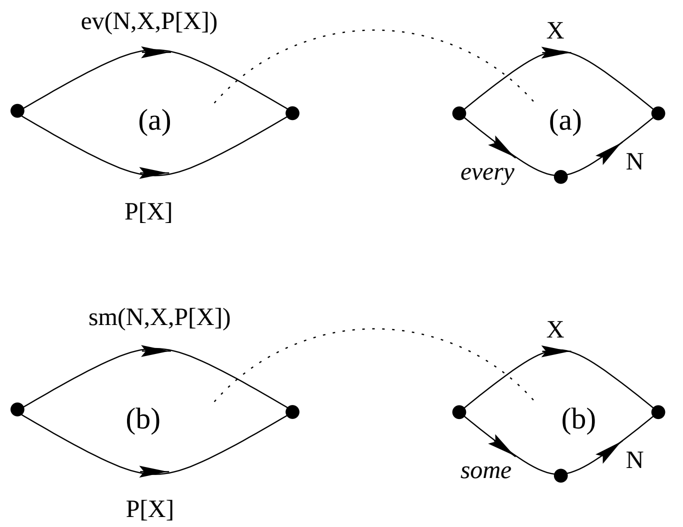


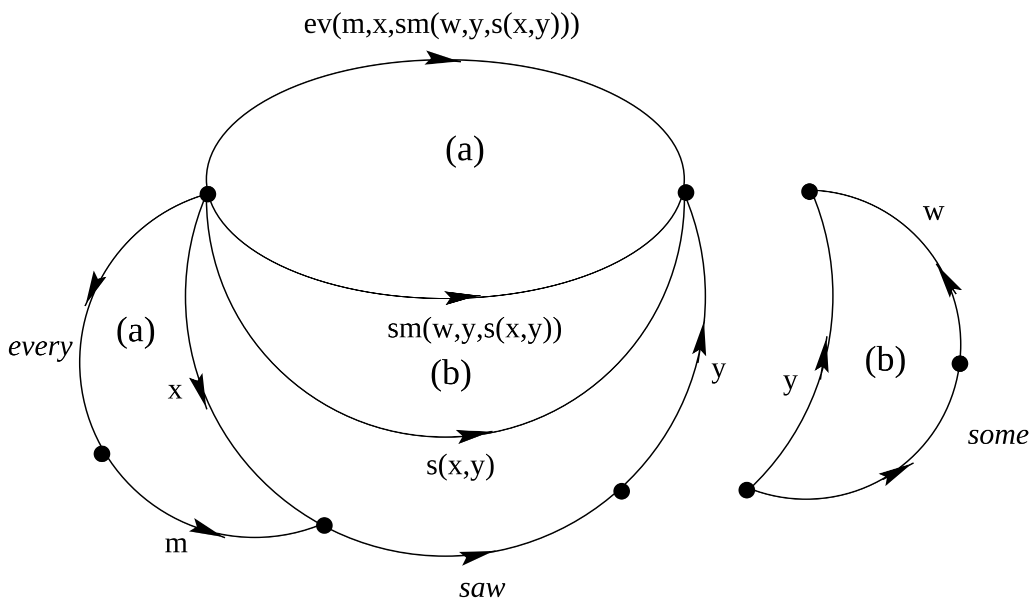
Multi-relators and non-boundedness: quantifier scoping



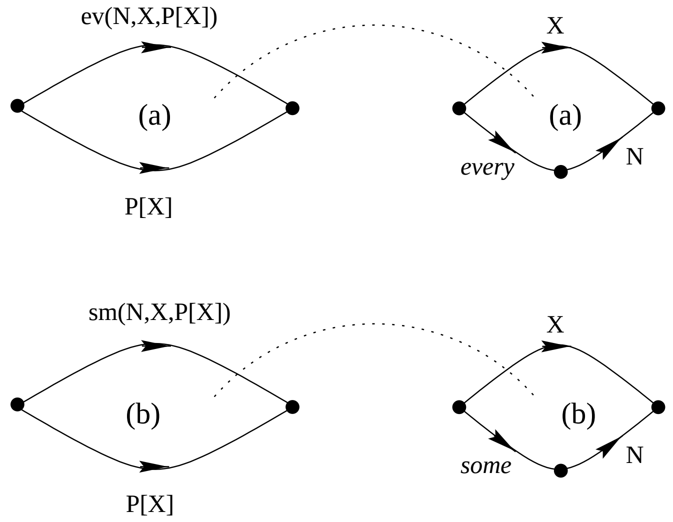


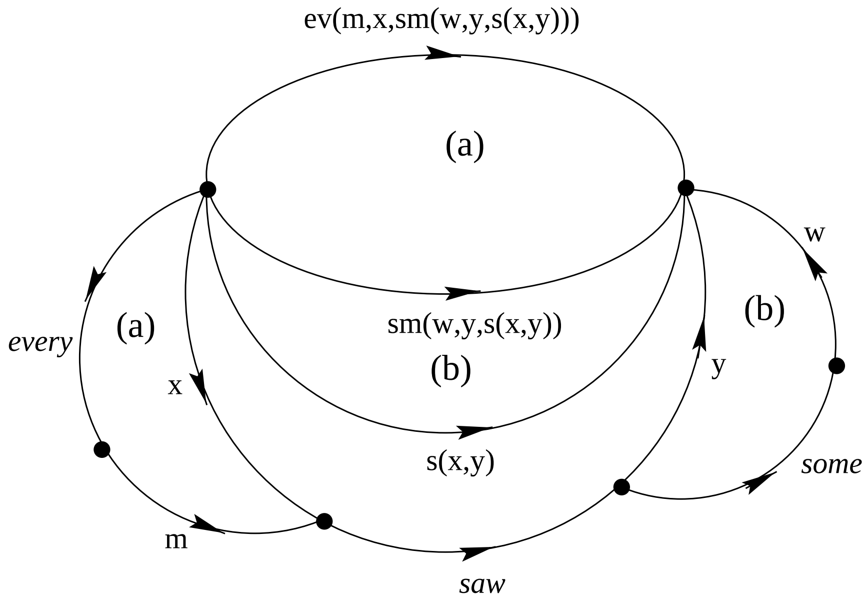
Multi-relators and non-boundedness: quantifier scoping



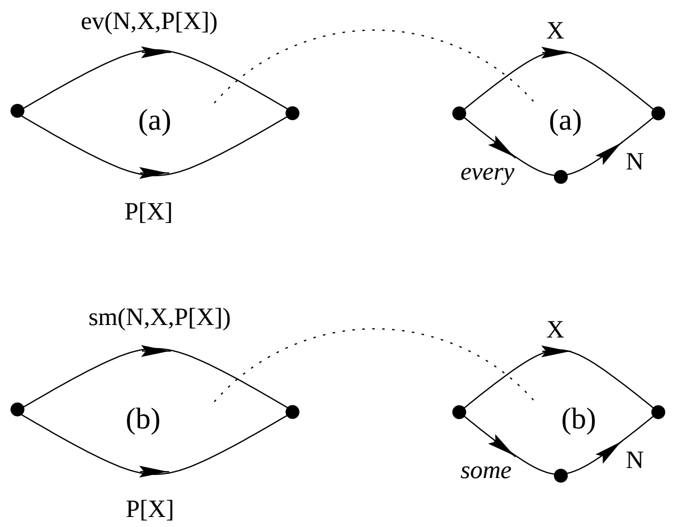


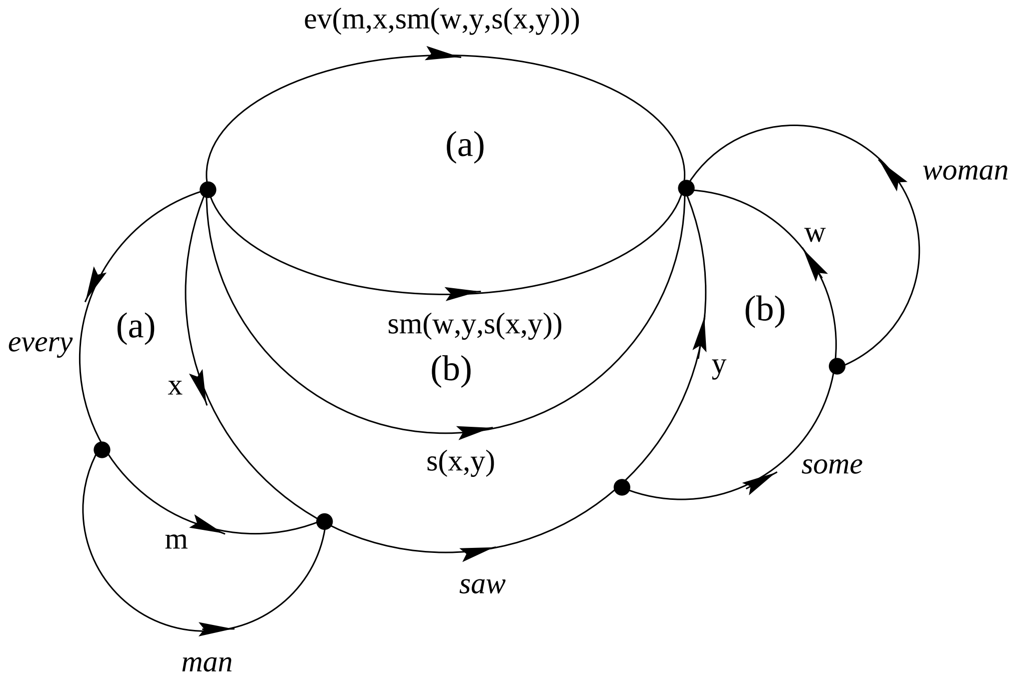
Multi-relators and non-boundedness: quantifier scoping



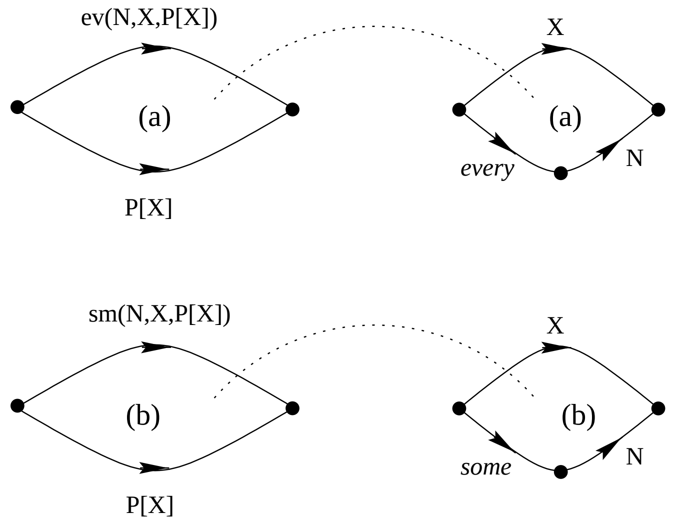


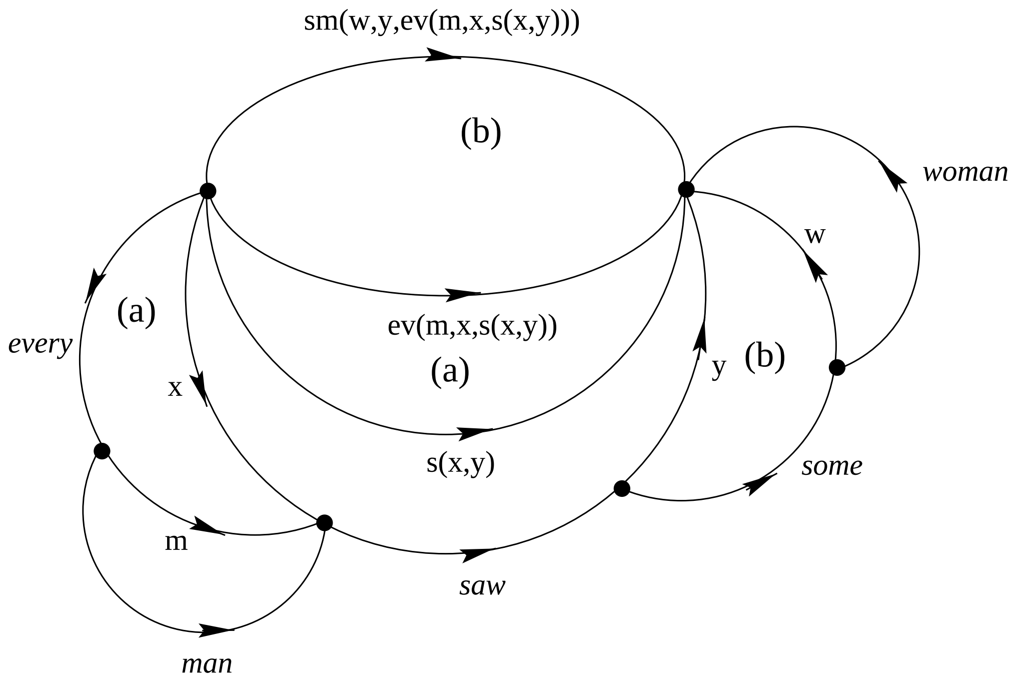
Multi-relators and non-boundedness: quantifier scoping



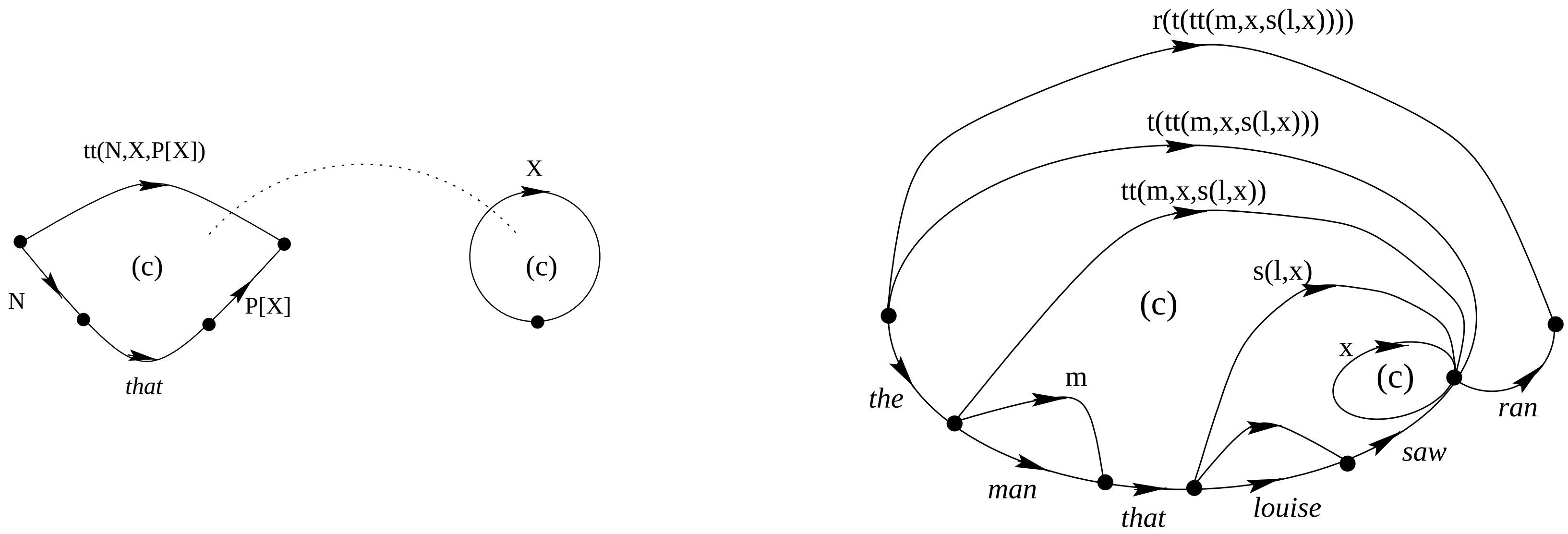


Multi-relators and non-boundedness: quantifier scoping (cont.)





Multi-relators and non-boundedness: relative clauses

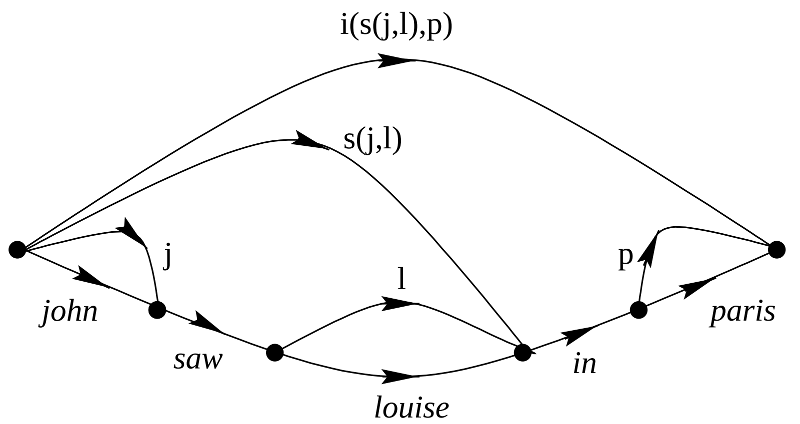


Complements

- Group computation and rewriting
- The use of group invariants for bounding computation complexity

Group computation and Rewriting: generation

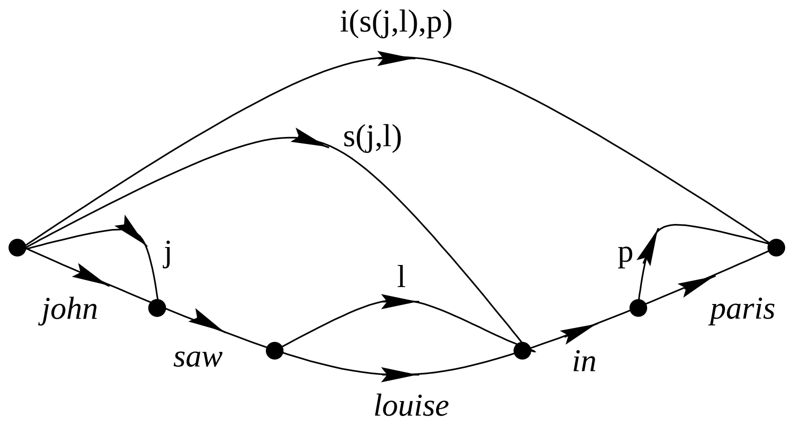
j ⇀ *john*
l ⇀ *louise*
p ⇀ *paris*
s(A,B) ⇀ A *saw* B
i(E,A) ⇀ E *in* A



i(s(j,l),p) ⇀ s(j,l) *in* p ⇀ j *saw* l *in* p ⇀ ... ⇀ *john saw louise in paris*

Group computation and Rewriting: parsing

$$\begin{aligned}
\textit{john} &\rightharpoondown \texttt{j} \\
\textit{louise} &\rightharpoondown \texttt{l} \\
\textit{paris} &\rightharpoondown \texttt{p} \\
\textit{saw} &\rightharpoondown \texttt{A}^{-1}\ \texttt{s(A,B)}\ \texttt{B}^{-1} \\
\textit{in} &\rightharpoondown \texttt{E}^{-1}\ \texttt{i(E,A)}\ \texttt{A}^{-1}
\end{aligned}$$



$$\textit{john saw louise in paris} \rightharpoondown \texttt{j}\ \textit{saw}\ \texttt{l}\ \textit{in}\ \texttt{p}$$

$$\rightharpoondown \texttt{j}\ \texttt{A}^{-1}\ \texttt{s(A,B)}\ \texttt{B}^{-1}\ \texttt{l}\ \texttt{E}^{-1}\ \texttt{i(E,A)}\ \texttt{A}^{-1}\ \texttt{p} \rightharpoondown \texttt{s(j,l)}\ \texttt{E}^{-1}\ \texttt{i(E,p)} \rightharpoondown \texttt{i(s(j,l),p)}$$

Group computation compared to rewriting

- Type-0 charts can be seen as diagrams over "bilateral" cells, that is cells whose boundary is of the form $\alpha\beta^{-1}$, for α, β positive words of $F(V)$.
- When does a diagram over bilateral cells correspond to a type-0 chart?
- Answer: exactly when the diagram is *acyclic* (see drawing).
- Consequence: If a rewriting system has no cycles, then its encoding as a group computation structure preserves the rewriting relation.
  - For CFGs, this means: finite language (a very restrictive requirement);
  - For DCGs (or for unification-based type-0 grammars), this means: well-foundedness of the generation process (a reasonable requirement).

Group computation compared to rewriting (cont.)

(a)	$\mathrm{s} \mapsto \mathrm{np}\ \mathrm{vp}$	(a')	$\mathrm{s}\ \mathrm{vp}^{-1}\mathrm{np}^{-1}$
	$\mathrm{np} \mapsto \mathit{john}$		$\mathrm{np}\ \mathit{john}^{-1}$
	$\mathrm{vp} \mapsto \mathit{walked}$		$\mathrm{vp}\ \mathit{walked}^{-1}$
	$\mathrm{vp} \mapsto \mathit{often}\ \mathrm{vp}$		$\mathrm{vp}\ \mathrm{vp}^{-1}\mathit{often}^{-1} = \mathit{often}^{-1}$
(b)	$\mathrm{s(.(NP,VP))} \mapsto \mathrm{np(NP)}\ \mathrm{vp(VP)}$	(b')	$\mathrm{s(.(NP,VP))}\ \mathrm{vp(VP)}^{-1}\,\mathrm{np(NP)}^{-1}$
	$\mathrm{np(john)} \mapsto \mathit{john}$		$\mathrm{np(john)}\ \mathit{john}^{-1}$
	$\mathrm{vp(walked)} \mapsto \mathit{walked}$		$\mathrm{vp(walked)}\ \mathit{walked}^{-1}$
	$\mathrm{vp(.(often,VP))} \mapsto \mathit{often}\ \mathrm{vp(VP)}$		$\mathrm{vp(.(often,VP))}\ \mathrm{vp(VP)}^{-1}\,\mathit{often}^{-1}$

Group invariants and the decidability of parsing/generation

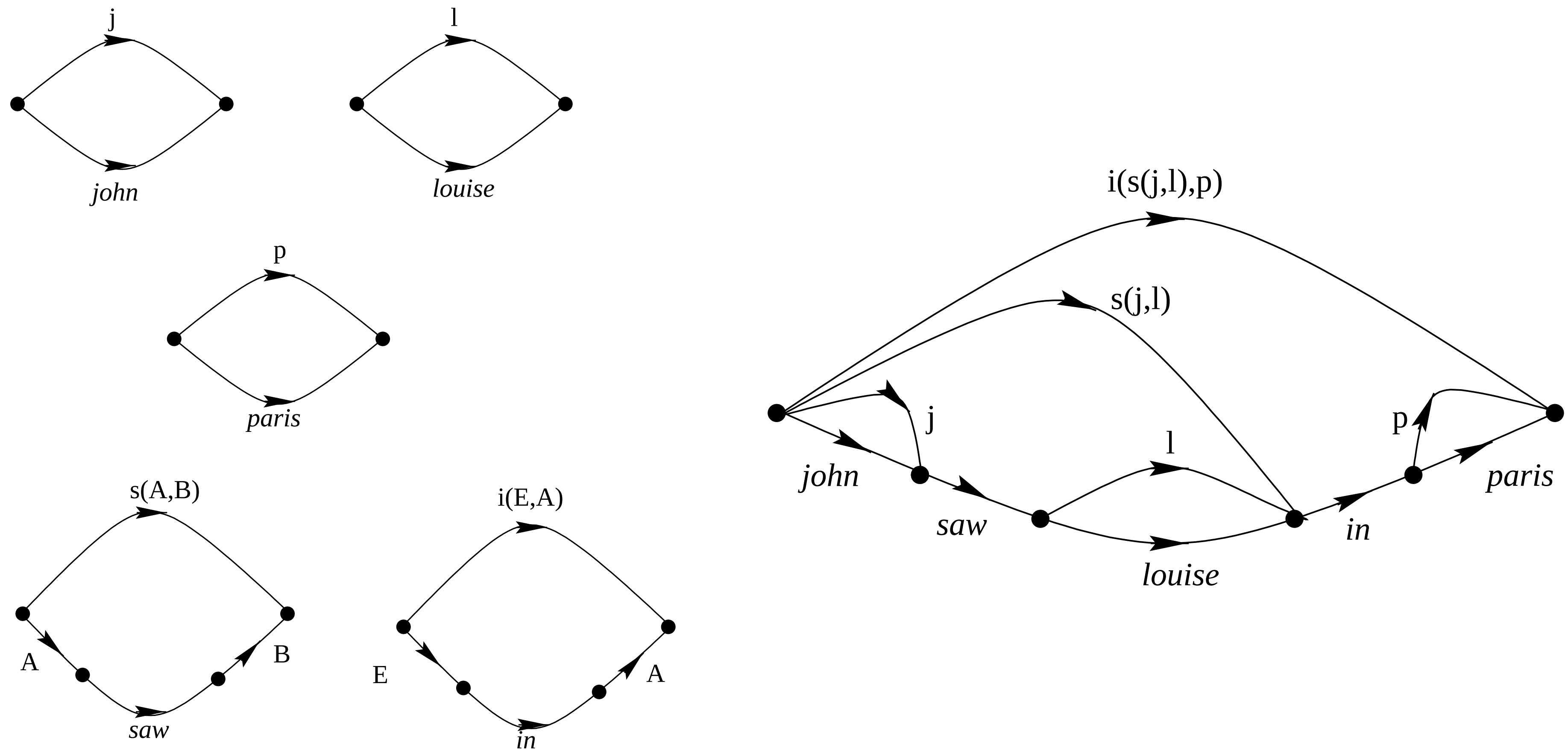
- Let ϕ be a morphism from $F(V)$ into $(\mathbb{R}, +)$, then, for any result $w = x_1 r_1 x_1{}^{-1} \cdots x_n r_n x_n{}^{-1}$, one has $\phi(w) = \sum_i \phi(r_i)$.

- If ϕ is such that, for all relators $r \in R$, and for some $k > 0$, one has $\phi(r) \geq k$, then the number n of cells of a diagram computing a result w is such that:

$$n \leq \phi(w)/k.$$

- This fact can be exploited for bounding the complexity of a computation:
  - Parsing: take ϕ_p to be a function extracting the "phonological content" of a result;
  - Generation: take ϕ_g to be a function extracting the "semantic content" of a result (see paper).

Group invariants and the decidability of parsing/generation (cont.)



References

M. Dymetman 1998: Group Theory and Computational Linguistics, Journal of Logic, Language and Information 7: 461-497, Kluwer

M. Dymetman 1999: Some Remarks on the Geometry of Grammar, in Proc. 6th Conference on the Mathematics of Language (MOL-6), Orlando, Florida

Also on: http://www.xrce.xerox.com/people/dymetman/dymetman.html