The String-Meaning Relations Definable by Lambek Grammars and Context-Free Grammars

Makoto Kanazawa and Sylvain Salvati

Lambek Grammars vs. Context-Free Grammars

- Expressivity w.r.t string sets
    - Bar-Hillel, Gaifman, and Shamir 1960, Cohen 1967
    - Pentus 1993 (cf. Chomsky 1963)
- Expressivity w.r.t string-meaning relations

They are known to be equivalent in terms of string-set generating power.
What about string-meaning relations?

Lambek Grammar

a	np/n	$\lambda uv.\exists x(ux \wedge vx)$
woman	n	$\lambda x.\textbf{woman}\ x$
who	(n\n)/(s/np)	$\lambda vux.ux \wedge vx$
John	np	$\lambda u.u\ \textbf{j}$
saw	(np\s)/np	$\lambda qp.p(\lambda x.q(\lambda y.\textbf{saw}\ y\ x))$
book	n	$\lambda x.\textbf{book}\ x$
gave	((np\s)/np)/np	$\lambda qrp.p(\lambda x.q(\lambda y.r(\lambda z.\textbf{gave}\ z\ y\ x)))$

⊢ np/n n (n\n)/(s/np) np (np\s)/np (np\s)/np np → s

a woman who John saw saw John

A Lambek grammar consists of triples of the form (terminal,type,lambda-term).
When a sequent is derivable, the string of terminals related to the antecedent types is generated by the grammar.
The meaning of the string is determined by the derivation.

Lambek Calculus

Initial sequents: $p \to p$

Rules:

$$\frac{\Pi \to A \quad \Gamma\, B\, \Delta \to C}{\Gamma\, \Pi\, A\backslash B\, \Delta \to C} \qquad \frac{A\, \Pi \to B}{\Pi \to A\backslash B}$$

$$\frac{\Pi \to A \quad \Gamma\, B\, \Delta \to C}{\Gamma\, B/A\, \Pi\, \Delta \to C} \qquad \frac{\Pi\, A \to B}{\Pi \to B/A}$$

$$\frac{\Pi \to C \quad \Gamma\, C\, \Delta \to A}{\Gamma\, \Pi\, \Delta \to A}\ \text{Cut}$$

The notion of derivation of sequents is given by this calculus.
Linear, non-commutative logic.

$$np \to np \qquad s \to s$$

$$np \to np \qquad np \;\; np\backslash s \to s$$

$$np \;\; (np\backslash s)/np \;\; np \to s \qquad n \to n \qquad n \to n$$

$$np \;\; (np\backslash s)/np \to s/np \qquad n \;\; n\backslash n \to n$$

$$n \;\; (n\backslash n)/(s/np) \;\; np \;\; (np\backslash s)/np \to n \qquad np \to np$$

$$np/n \;\; n \;\; (n\backslash n)/(s/np) \;\; np \;\; (np\backslash s)/np \to np \qquad s \to s$$

$$np \to np \qquad np/n \;\; n \;\; (n\backslash n)/(s/np) \;\; np \;\; (np\backslash s)/np \;\; np\backslash s \to s$$

$$np/n \;\; n \;\; (n\backslash n)/(s/np) \;\; np \;\; (np\backslash s)/np \;\; (np\backslash s)/np \;\; np \to s$$

a woman who John saw saw John $\in L(G)$

An example of a derivation.

Curry-Howard Homomorphism

$$A_1 \ldots A_n \rightarrow M[x_1,\ldots,x_n]{:}B$$

Initial sequents: $p \rightarrow x_1{:}p$

Rules:

$$\frac{\Pi \rightarrow N[\ldots]{:}A \quad \Gamma\, B\, \Delta \rightarrow M[\ldots,x_i,\ldots]{:}C}{\Gamma\, B/A\, \Pi\, \Delta \rightarrow M[\ldots,x_i\, N[\ldots],\ldots]{:}C}$$

$$\frac{\Pi\, A \rightarrow M[\ldots,x_n]{:}B}{\Pi \rightarrow \lambda z.M[\ldots,z]{:}B/A}$$

$$\frac{\Pi \rightarrow N[\ldots]{:}C \quad \Gamma\, C\, \Delta \rightarrow M[\ldots,x_i,\ldots]{:}A}{\Gamma\, \Pi\, \Delta \rightarrow M[\ldots,N[\ldots],\ldots]{:}B}\ \text{Cut}$$

A derivation determines a corresponding typed lambda-term.

$$\dfrac{np \rightarrow np \qquad s \rightarrow s}{np \;\; np\backslash s \rightarrow s}$$

$$\dfrac{np \rightarrow np \qquad np \;\; np\backslash s \rightarrow s}{np \;\; (np\backslash s)/np \;\; np \rightarrow s}$$

$$\dfrac{np \;\; (np\backslash s)/np \;\; np \rightarrow s}{np \;\; (np\backslash s)/np \rightarrow s/np} \qquad \dfrac{n \rightarrow n \qquad n \rightarrow n}{n \;\; n\backslash n \rightarrow n}$$

$$\dfrac{np \;\; (np\backslash s)/np \rightarrow s/np \qquad n \;\; n\backslash n \rightarrow n}{n \;\; (n\backslash n)/(s/np) \;\; np \;\; (np\backslash s)/np \rightarrow n}$$

$$\dfrac{n \;\; (n\backslash n)/(s/np) \;\; np \;\; (np\backslash s)/np \rightarrow n \qquad np \rightarrow np}{np/n \;\; n \;\; (n\backslash n)/(s/np) \;\; np \;\; (np\backslash s)/np \rightarrow np}$$

$$\dfrac{np/n \;\; n \;\; (n\backslash n)/(s/np) \;\; np \;\; (np\backslash s)/np \rightarrow np \qquad s \rightarrow s}{np/n \;\; n \;\; (n\backslash n)/(s/np) \;\; np \;\; (np\backslash s)/np \;\; np\backslash s \rightarrow s}$$

$$\dfrac{np \rightarrow np \qquad np/n \;\; n \;\; (n\backslash n)/(s/np) \;\; np \;\; (np\backslash s)/np \;\; np\backslash s \rightarrow s}{np/n \;\; n \;\; (n\backslash n)/(s/np) \;\; np \;\; (np\backslash s)/np \;\; (np\backslash s)/np \;\; np \rightarrow x_6 x_7 (x_1 (x_3 (\lambda z. x_5 z x_4)) x_2) : s}$$

a woman who John saw saw John

$$x_6 x_7 (x_1 (x_3 (\lambda z. x_5 z x_4)) x_2) \begin{bmatrix} x_1 := \lambda uv. \exists x (ux \wedge vx) \\ x_2 := \lambda x. \mathbf{woman}\; x \\ x_3 := \lambda vux. ux \wedge vx \\ x_4 := \lambda u.u\; \mathbf{j} \\ x_5 := \lambda qp.p(\lambda x.q(\lambda y. \mathbf{saw}\; y\; x)) \\ x_6 := \lambda qp.p(\lambda x.q(\lambda y. \mathbf{saw}\; y\; x)) \\ x_7 := \lambda u.u\; \mathbf{j} \end{bmatrix}$$

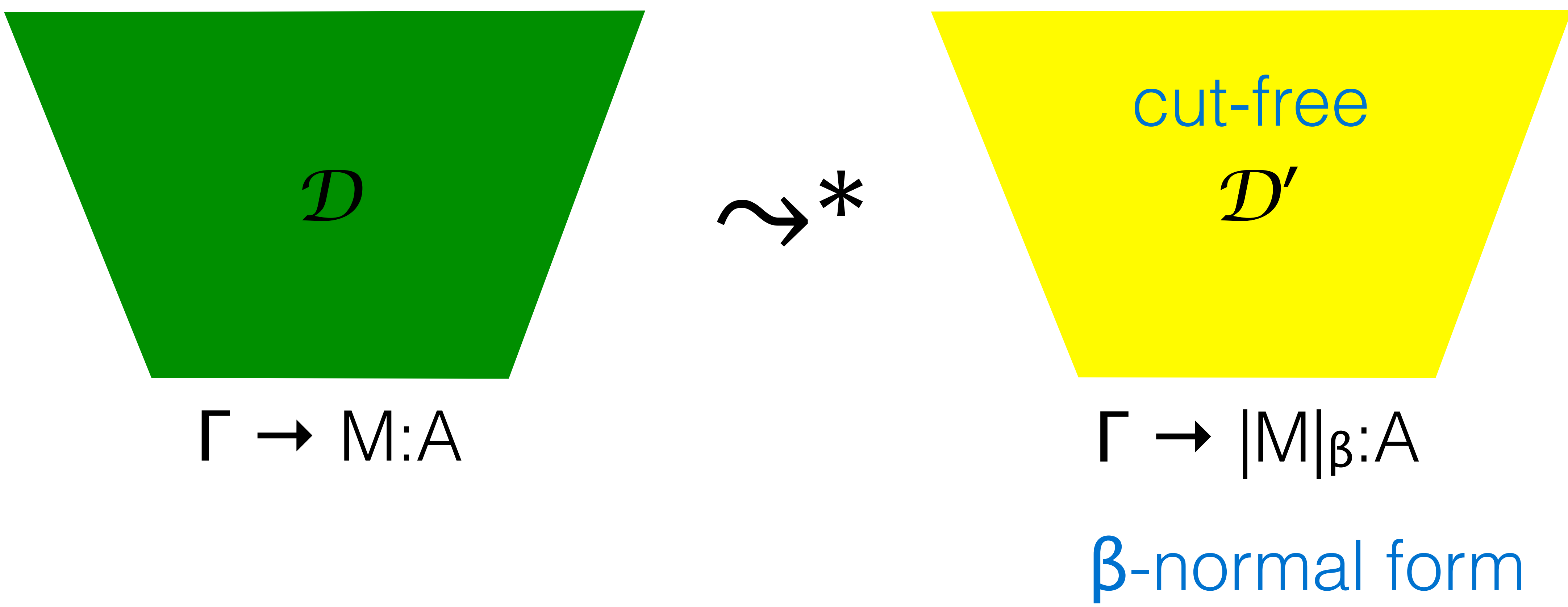
$$\twoheadrightarrow \exists x ((\mathbf{woman}\; x \wedge \mathbf{saw}\; x\; \mathbf{j}) \wedge \mathbf{saw}\; \mathbf{j}\; x)$$

(a woman who John saw saw John, $\exists x ((\mathbf{woman}\; x \wedge \mathbf{saw}\; x\; \mathbf{j}) \wedge \mathbf{saw}\; \mathbf{j}\; x)) \in R(G)$

An example of a string-lambda-term pair generated by the grammar.

Cut Elimination

- Lambek 1958



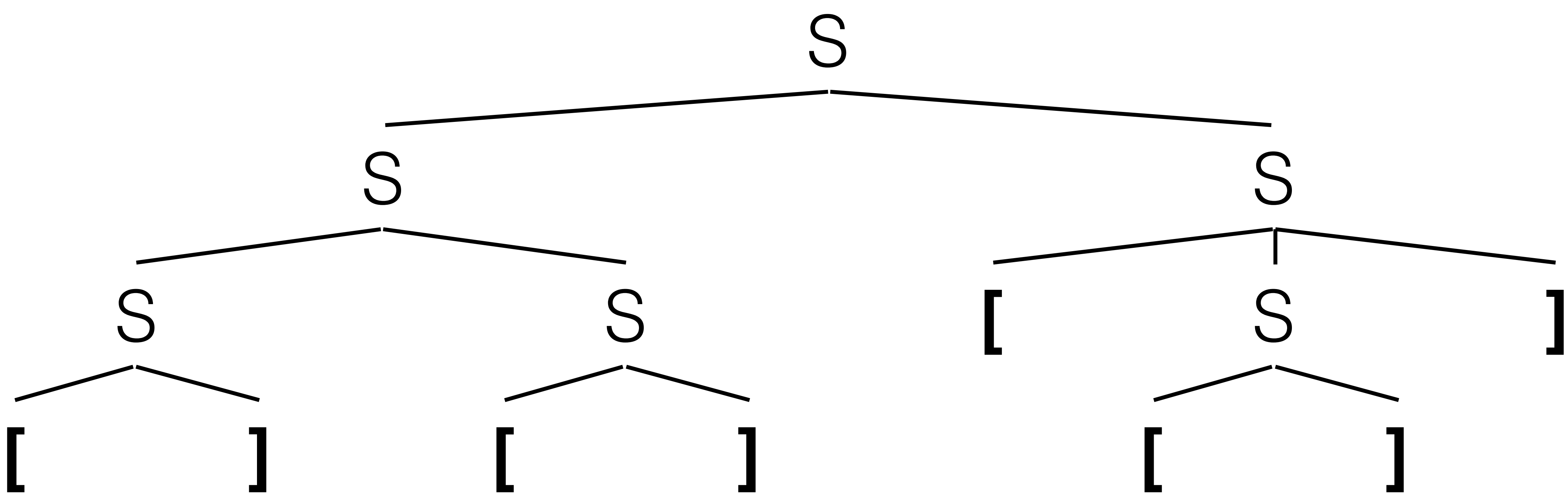
An important fact about sequent calculus.

Context-Free Grammar with Montague Semantics

$S \to \mathbf{[\,]} \;:\; \ulcorner 1 \urcorner \;\; (= \lambda fy.fy)$

$S \to \mathbf{[}\,S\,\mathbf{]} \;:\; \mathrm{suc}\; x_1 \;\; (= \lambda fy.x_1 f(fy))$

$S \to S\,S \;:\; \mathrm{plus}\; x_1\; x_2 \;\; (= \lambda fy.x_1 f(x_2 fy))$



$L(G) = D_1 - \{ \varepsilon \}$

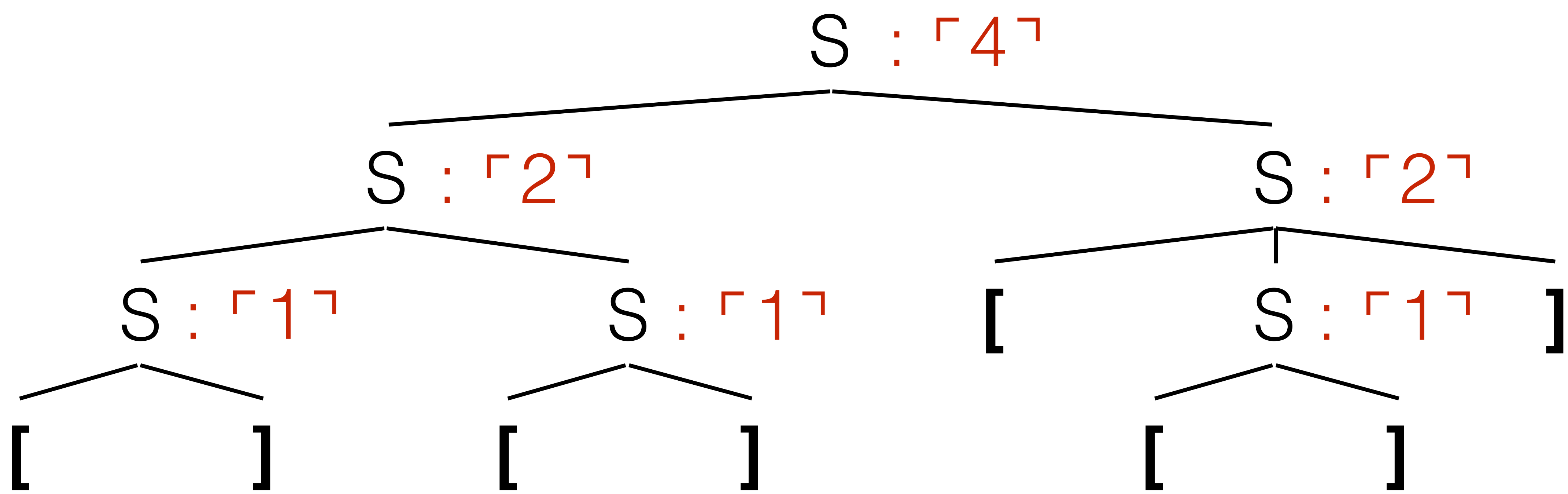
Context-free grammars with semantics expressed with typed lambda-terms.
Each rule is associated with a lambda-term with n free variables, where n is the number of nonterminals on the right-hand side.

Context-Free Grammar with Montague Semantics

$$S \rightarrow \mathbf{[\ ]} : \ulcorner 1 \urcorner \quad (= \boldsymbol{\lambda} fy.fy)$$

$$S \rightarrow \mathbf{[}\ S\ \mathbf{]} : \text{suc}\ x_1 \quad (= \boldsymbol{\lambda} fy.x_1 f(fy))$$

$$S \rightarrow S\ S : \text{plus}\ x_1\ x_2 \quad (= \boldsymbol{\lambda} fy.x_1 f(x_2 fy))$$



$$R(G) = \{\ (w, \ulcorner |w|_{\mathbf{[}} \urcorner) \mid w \in D_1 - \{\ \boldsymbol{\varepsilon}\ \}\ \}$$

How the meaning is determined by derivation.

Lambek

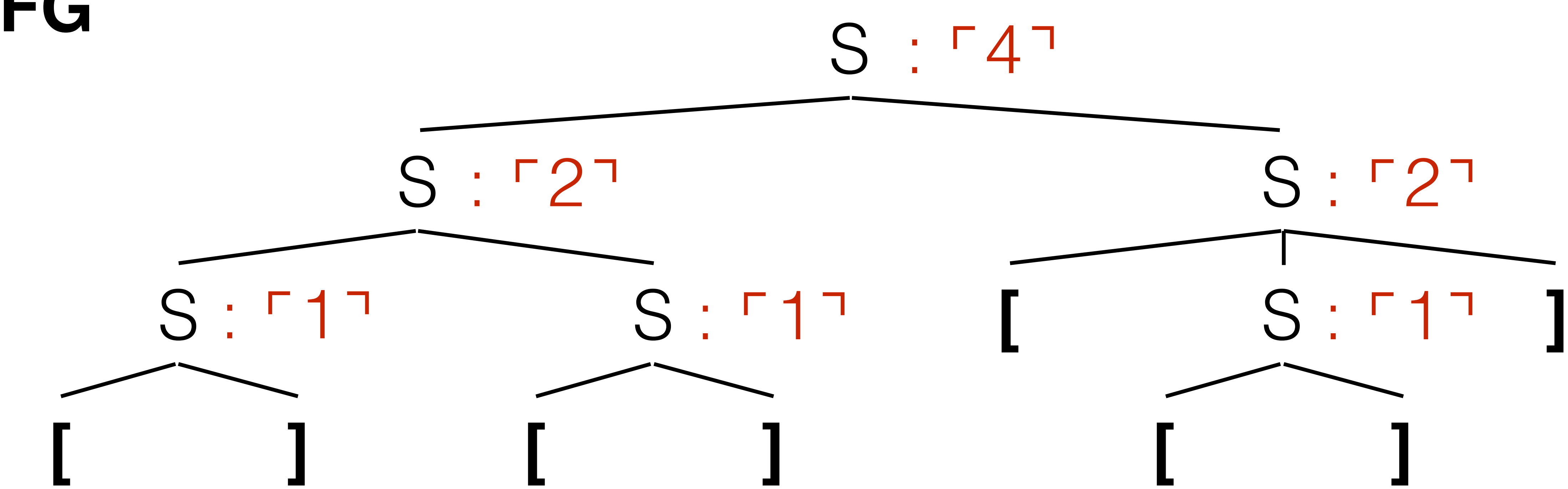
$$\dfrac{\dfrac{np \to np \qquad \dfrac{\dfrac{np \to np \qquad \dfrac{np \to np \quad s \to s}{np\ \ np\backslash s \to s}}{np\ \ (np\backslash s)/np\ \ np \to s}}{np\ \ (np\backslash s)/np \to s/np} \quad \dfrac{n \to n \quad n \to n}{n\ \ n\backslash n \to n}}{n\ \ (n\backslash n)/(s/np)\ \ np\ \ (np\backslash s)/np \to n} \quad np \to np}{np/n\ \ n\ \ (n\backslash n)/(s/np)\ \ np\ \ (np\backslash s)/np \to np}$$

$$\dfrac{np \to np \qquad \dfrac{np/n\ \ n\ \ (n\backslash n)/(s/np)\ \ np\ \ (np\backslash s)/np \to np \qquad s \to s}{np/n\ \ n\ \ (n\backslash n)/(s/np)\ \ np\ \ (np\backslash s)/np\ \ np\backslash s \to s}}{np/n\ \ n\ \ (n\backslash n)/(s/np)\ \ np\ \ (np\backslash s)/np\ \ (np\backslash s)/np\ \ np \to x_6 x_7(x_1(x_3(\lambda z.x_5 z x_4))x_2):s}$$

a woman who John saw saw John

$$\left[\begin{array}{l} x_1 := \lambda uv.\exists x(ux \wedge vx) \\ x_2 := \lambda x.\mathbf{woman}\ x \\ x_3 := \lambda vux.ux \wedge vx \\ x_4 := \lambda u.u\ \mathbf{j} \\ x_5 := \lambda qp.p(\lambda x.q(\lambda y.\mathbf{saw}\ y\ x)) \\ x_6 := \lambda qp.p(\lambda x.q(\lambda y.\mathbf{saw}\ y\ x)) \\ x_7 := \lambda u.u\ \mathbf{j} \end{array}\right]$$

CFG



Two formalisms to be compared.

CFG → Lambek

- Greibach normal form

$$A \rightarrow a\ B_1 \ldots B_n : M[x_1, \ldots x_n]$$

⬇

$$a : (\ldots(A/B_n)/\ldots)/B_1 : \lambda x_1 \ldots x_n . M[x_1, \ldots x_n]$$

AB grammar

Conversion from CFG to Lambek grammar is based on Greibach normal form.
Once Greibach normal form is obtained, one can easily get an AB grammar, which is equivalent to a Lambek grammar.

Greibach Normal Form Transformation

cycle-free CFG G s.t. **ε** ∉ L(G)

↓ elimination of **ε**-rules

↓ elimination of unit-rules

↓ left-corner transform

↓

extended Greibach normal form

↓

Greibach normal form

Can use left-corner transform to perform Greibach normal form transformation.
Grammar need to be cycle-free (finitely ambiguous).

Left-corner Transform

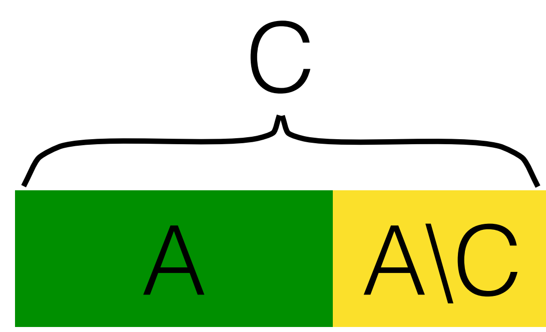
$A \rightarrow a\, \alpha \quad : M[\ldots]$

$A \rightarrow B\, \beta \quad : N[x_1, \ldots]$

$\Downarrow$

$A \rightarrow a\, \alpha \quad : M[\ldots]$

$C \rightarrow a\, \alpha\ A\backslash C \quad : x_n\, M[\ldots]$



$B\backslash A \rightarrow \beta \quad : \lambda z.N[z, \ldots]$

$B\backslash C \rightarrow \beta\ A\backslash C \quad : \lambda z.x_n\, N[z, \ldots]$

Introduce new nonterminals of the form A\C, whose meaning is a function from A-meanings to C-meanings.

$S \rightarrow \mathbf{[\ ]} \ : \ \ulcorner 1 \urcorner$

$S \rightarrow \mathbf{[}\ S\ \mathbf{]} \ : \ \mathrm{suc}\ x_1$

$S \rightarrow S\ S \ : \ \mathrm{plus}\ x_1\ x_2$

left-corner transform

$S \rightarrow \mathbf{[\ ]} \ : \ \ulcorner 1 \urcorner$

$S \rightarrow \mathbf{[\ ]}\ S\backslash S \ : \ x_1\ \ulcorner 1 \urcorner$

$S \rightarrow \mathbf{[}\ S\ \mathbf{]} \ : \ \mathrm{suc}\ x1$

$S \rightarrow \mathbf{[}\ S\ \mathbf{]}\ S\backslash S \ : \ x_2\ (\mathrm{suc}\ x_1)$

$S\backslash S \rightarrow S \ : \ \lambda z.\mathrm{plus}\ z\ x_1$

$S\backslash S \rightarrow S\ S\backslash S \ : \ \lambda z.x_2\ (\mathrm{plus}\ z\ x_1)$

left-corner transform

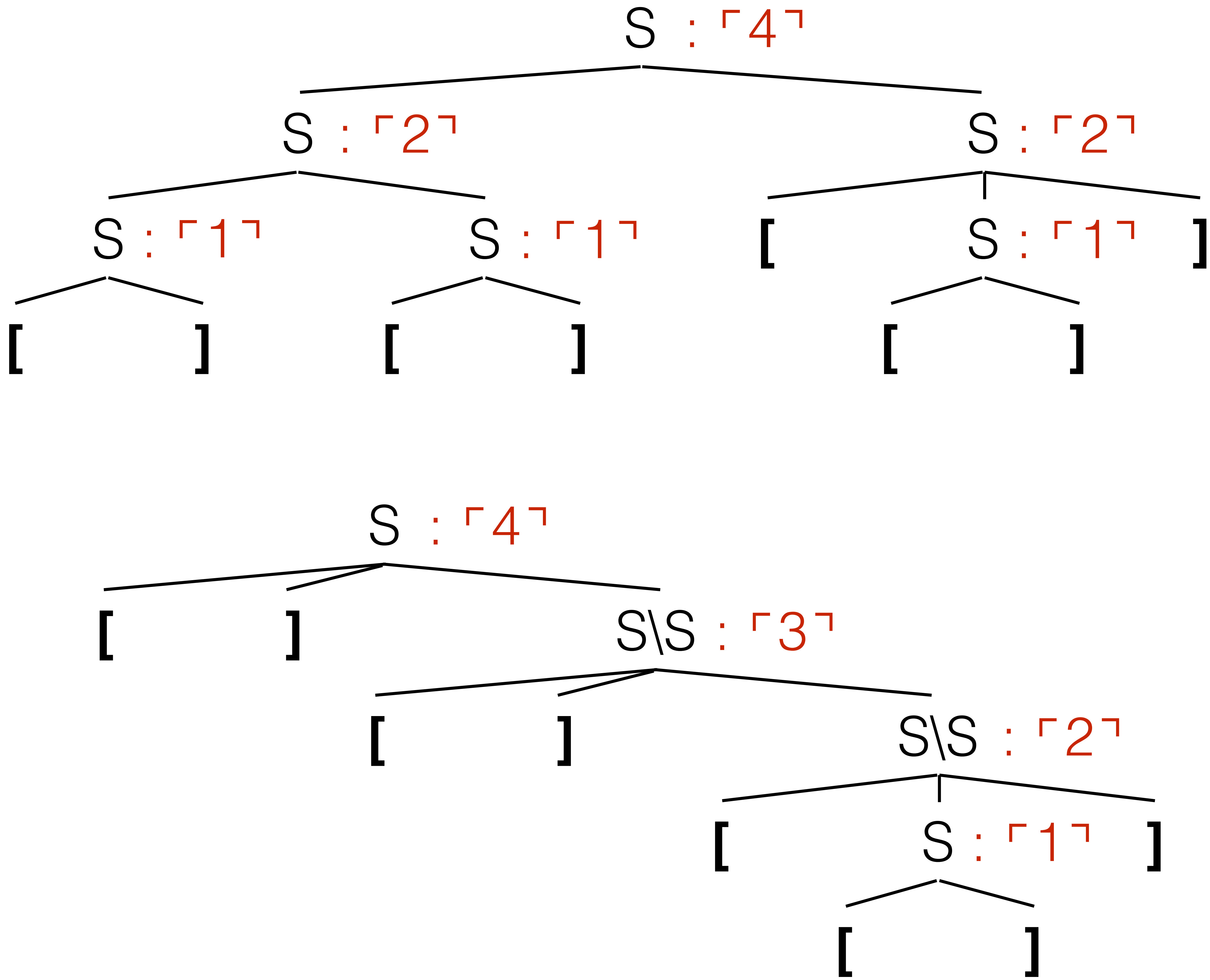
$$
\begin{aligned}
S &\to \mathbf{[\ ]} &&: \ulcorner 1 \urcorner \\
S &\to \mathbf{[\ ]}\ S\backslash S &&: x_1\ \ulcorner 1 \urcorner \\
S &\to \mathbf{[}\ S\ \mathbf{]} &&: \text{suc}\ x_1 \\
S &\to \mathbf{[}\ S\ \mathbf{]}\ S\backslash S &&: x_2\,(\text{suc}\ x_1) \\
S\backslash S &\to S &&: \lambda z.\text{plus}\ z\ x_1 \\
S\backslash S &\to S\ S\backslash S &&: \lambda z.x_2\,(\text{plus}\ z\ x_1)
\end{aligned}
$$

unfolding

$$
\begin{aligned}
&\vdots && \vdots \\
S\backslash S &\to \mathbf{[\ ]} &&: \lambda z.\text{plus}\ z\ \ulcorner 1 \urcorner \\
S\backslash S &\to \mathbf{[\ ]}\ S\backslash S &&: \lambda z.\text{plus}\ z\ (x_1\ \ulcorner 1 \urcorner) \\
S\backslash S &\to \mathbf{[}\ S\ \mathbf{]} &&: \lambda z.\text{plus}\ z\ (\text{suc}\ x_1) \\
S\backslash S &\to \mathbf{[}\ S\ \mathbf{]}\ S\backslash S &&: \lambda z.\text{plus}\ z\ (x_2\,(\text{suc}\ x_1)) \\
S\backslash S &\to \mathbf{[\ ]}\ S\backslash S &&: \lambda z.x_1\,(\text{plus}\ z\ \ulcorner 1 \urcorner) \\
&\vdots && \vdots
\end{aligned}
$$

extended Greibach normal form

Expanding the rightmost nonterminal on the right-hand side gives extended Greibach normal form (grammar where the right-hand side of each rule starts with a terminal).



How derivation trees are transformed.
Conversion from extended Greibach normal form to Greibach normal form is easy.

Lambek → CFG

- Pentus 1993, 1997
  - interpolation
  - combinatorial properties of Lambek calculus derivations

Standard properties of sequent calculus and combinatorial properties crucially dependent on non-commutativity are involved.

Lambek grammar G

m = maximal size of types used in G

$S_G = \{ A_1 \ldots A_n \to A_{n+1} \mid$

$\vdash A_1 \ldots A_n \to A_{n+1}$

$\mathrm{size}(A_i) \leq m \ (1 \leq i \leq n+1)$

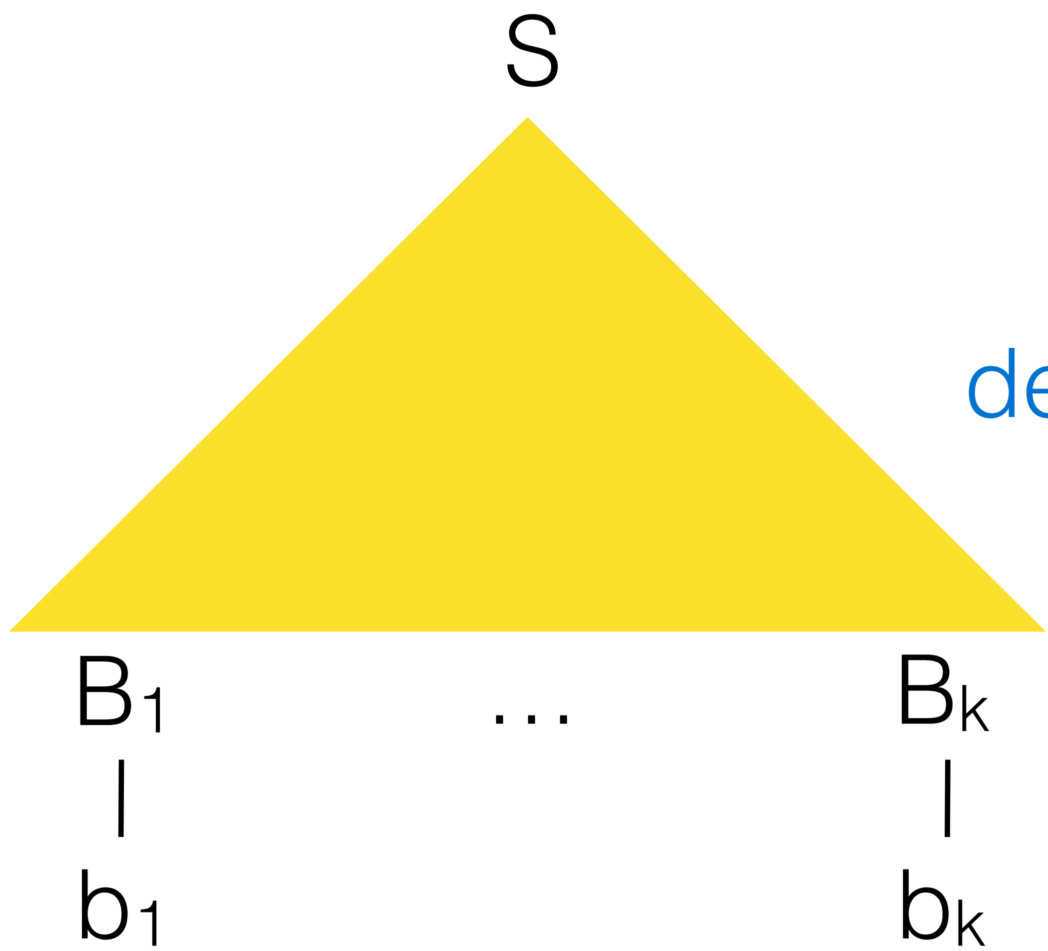
$\mathrm{size}(A_1) + \cdots + \mathrm{size}(A_n) \leq 2m \}$

CFG G′

$A_{n+1} \to A_1 \ldots A_n \qquad (A_1 \ldots A_n \to A_{n+1} \in S_G)$

$A \to a \qquad$ (a:A is in G)

Let’s see how it goes in the absence of semantics.
The construction of a CFG equivalent to a given Lambek grammar is easy to describe.



derivation in G′

$$S_G \vdash_{Cut} B_1 \dots B_k \to S$$

$$\frac{\Pi \to C \quad \Gamma\, C\, \Delta \to A}{\Gamma\, \Pi\, \Delta \to A}\ \text{Cut}$$

$$\vdash B_1 \dots B_k \to S$$

$L(G') \subseteq L(G)$ ✓

Prove equivalence of the two grammars.
One direction is easy.

Interpolation

$\vdash \Gamma\ \Pi\ \Delta \rightarrow A$

$\vdash \Pi \rightarrow C \qquad \vdash \Gamma\ C\ \Delta \rightarrow A$

$$\frac{\Pi \rightarrow C \quad \Gamma\ C\ \Delta \rightarrow A}{\Gamma\ \Pi\ \Delta \rightarrow A}\ \text{Cut}$$

C: interpolant

- C is “small”
- Π must be “connected”

Interpolation is used to prove the other direction.
Given a derivable sequent and a subsequence of its antecedent, can get two derivable sequents that combine with Cut to give the original sequent.
The interpolant C satisfies a special property.
In implicational logics, one in general gets a sequence of formulas, rather than a single formula.

$$np \rightarrow np \qquad s \rightarrow s$$

$$\overline{}$$

$$np \rightarrow np \qquad np \;\; np\backslash s \rightarrow s$$

$$\overline{}$$

$$np \;\; (np\backslash s)/np \;\; np \rightarrow s \qquad n \rightarrow n \qquad n \rightarrow n$$

$$np \;\; (np\backslash s)/np \rightarrow s/np \qquad n \;\; n\backslash n \rightarrow n$$

$$\overline{}$$

$$n \;\; (n\backslash n)/(s/np) \;\; np \;\; (np\backslash s)/np \rightarrow n \qquad np \rightarrow np$$

$$\overline{}$$

$$np/n \;\; n \;\; (n\backslash n)/(s/np) \;\; np \;\; (np\backslash s)/np \rightarrow np \qquad s \rightarrow s$$

$$\overline{}$$

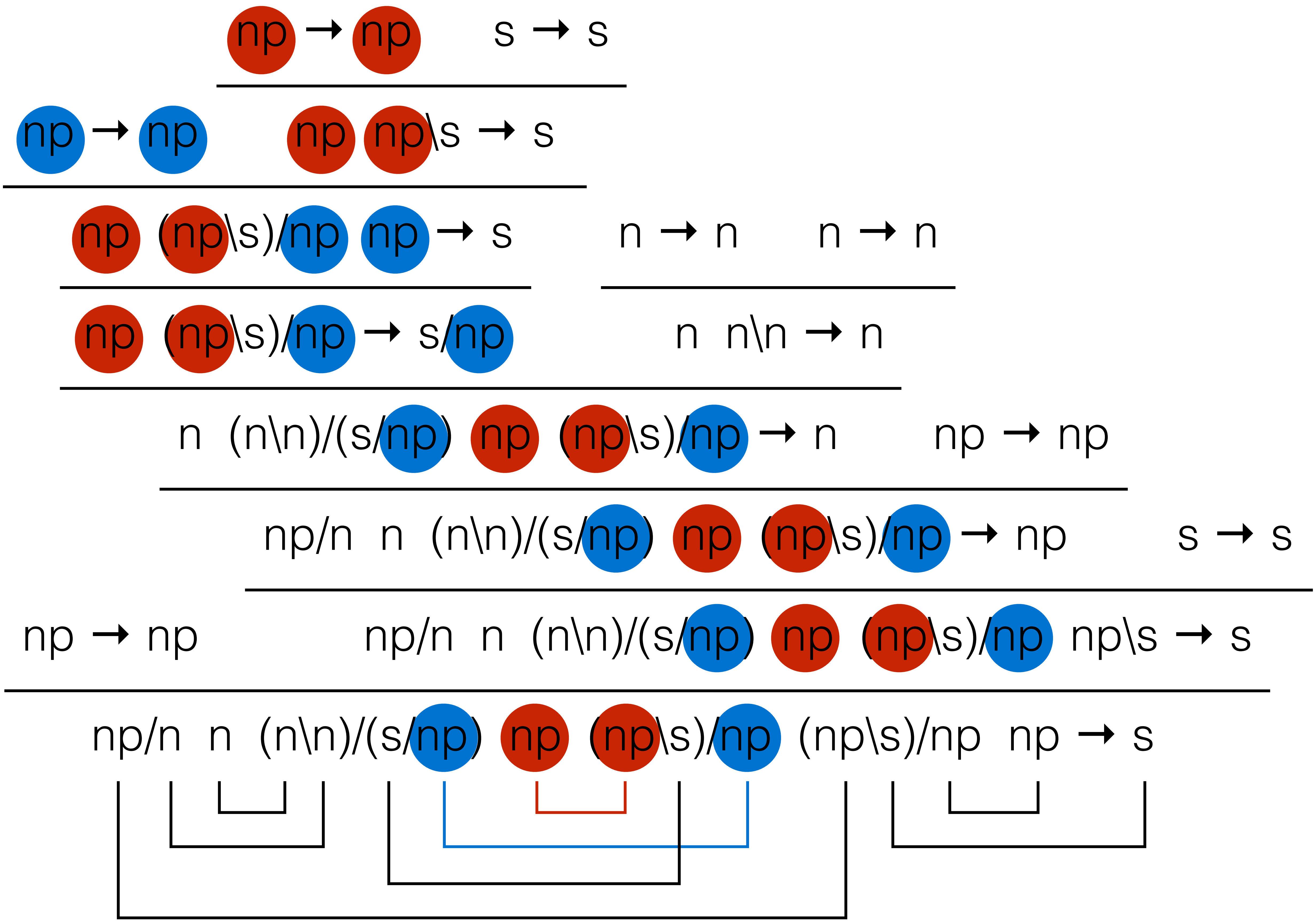
$$np \rightarrow np \qquad np/n \;\; n \;\; (n\backslash n)/(s/np) \;\; np \;\; (np\backslash s)/np \;\; np\backslash s \rightarrow s$$

$$\overline{}$$

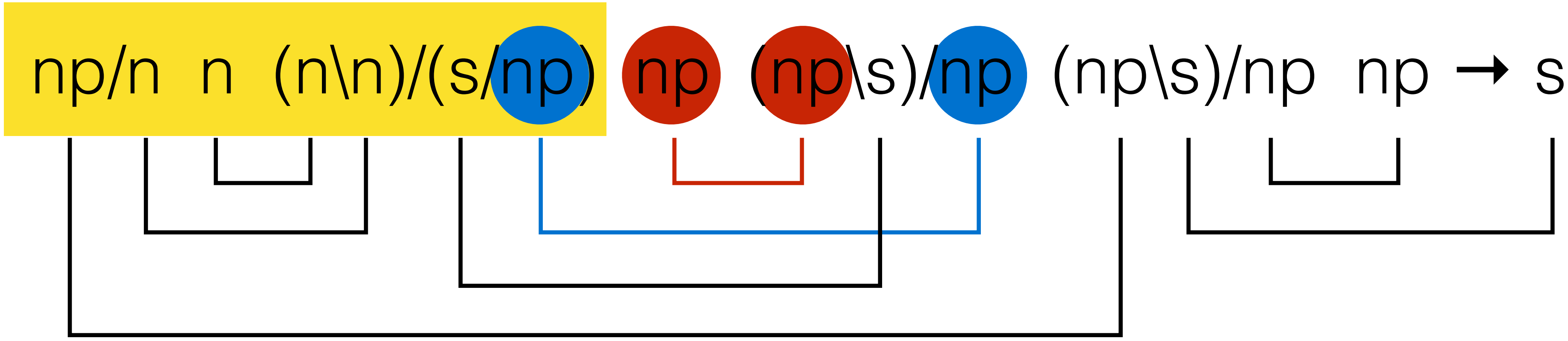
$$np/n \;\; n \;\; (n\backslash n)/(s/np) \;\; np \;\; (np\backslash s)/np \;\; (np\backslash s)/np \;\; np \rightarrow s$$

a woman who John saw saw John $\in L(G)$

An example illustrating the condition on the interpolant.



Pairing between positive and negative occurrences of atomic types.



np/n n (n\n)/(s/np) → np/(s/np)

np/(s/np) np (np\s)/np (np\s)/np np → s

size(np/(s/np)) = # outgoing links from np/n n (n\n)/(s/np)

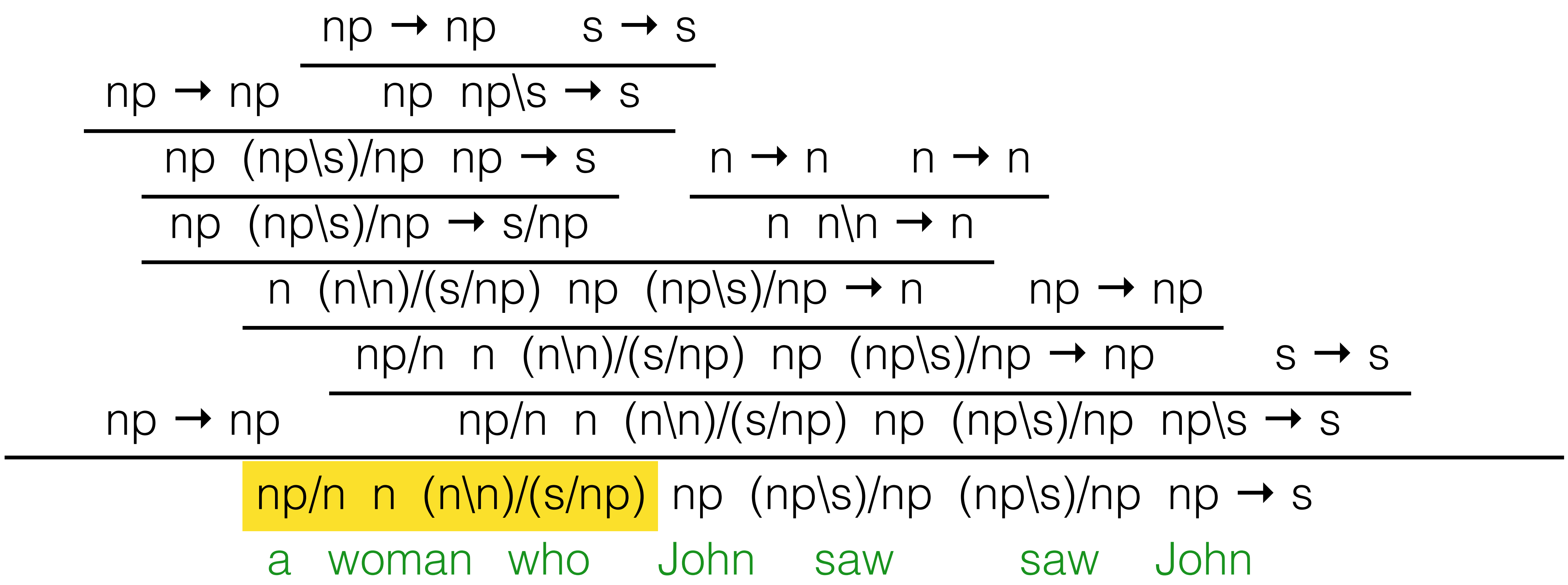
Interpolation

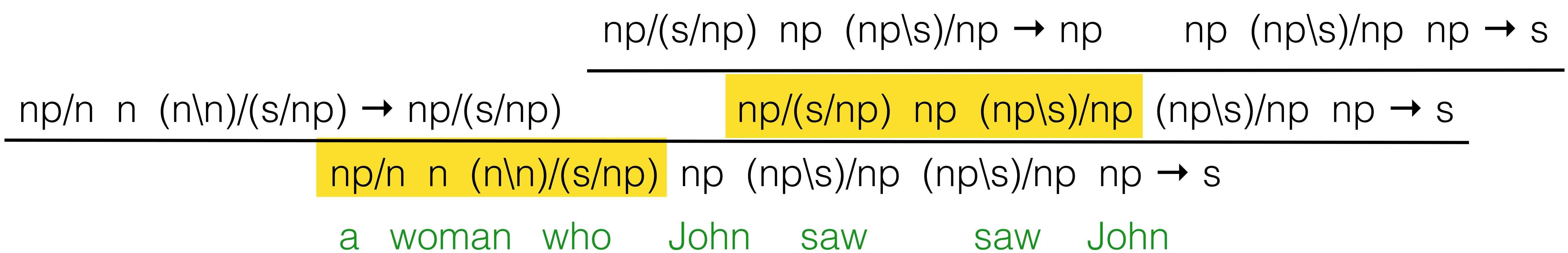
$\vdash \Gamma\ \Pi\ \Delta \rightarrow A$

$\vdash \Pi \rightarrow C \qquad \vdash \Gamma\ C\ \Delta \rightarrow A$

$$\frac{\Pi \rightarrow C \quad \Gamma\ C\ \Delta \rightarrow A}{\Gamma\ \Pi\ \Delta \rightarrow A}\ \text{Cut}$$

- C is "small" size(C) = # of outgoing links from Π
- Π must be "connected"





Applying interpolation repeatedly, obtain derivations from simple sequents using Cut only.

np/(s/np) np (np\s)/np ➝ np   np (np\s)/np np ➝ s

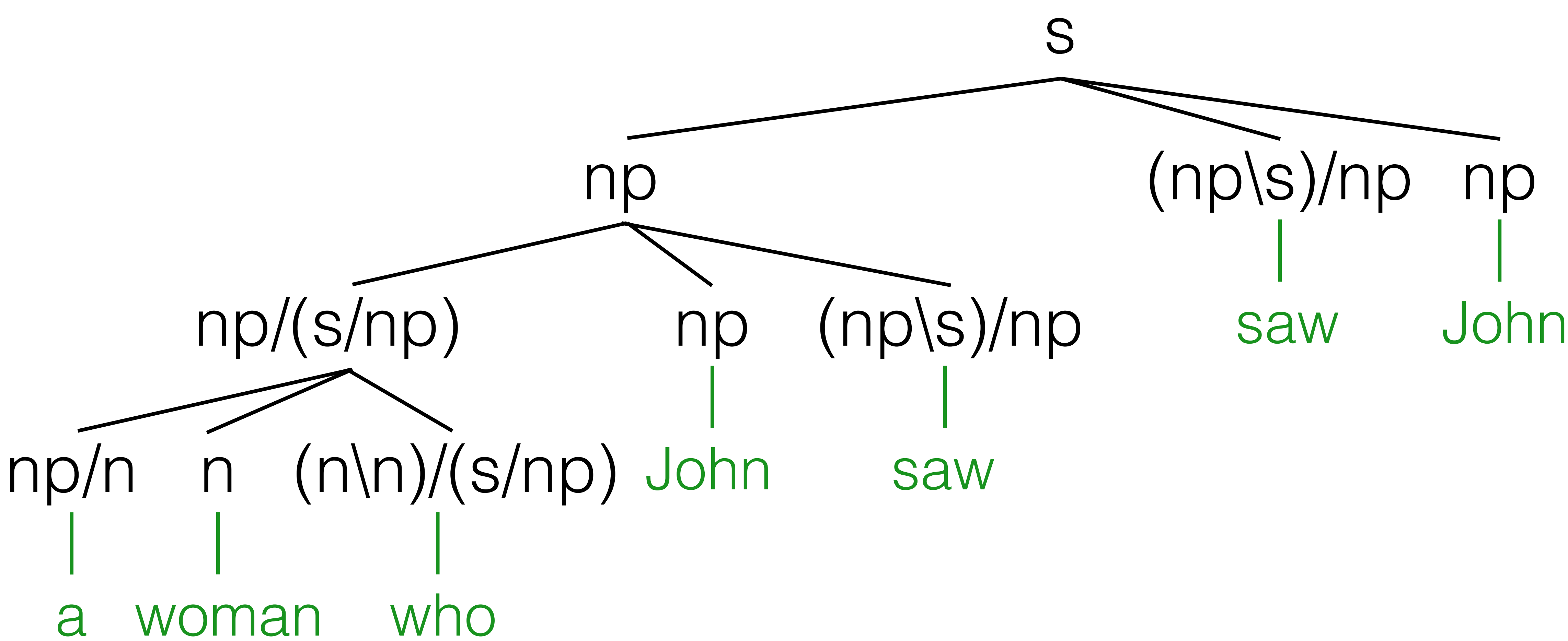
---

np/n n (n\n)/(s/np) ➝ np/(s/np)   np/(s/np) np (np\s)/np (np\s)/np np ➝ s

---

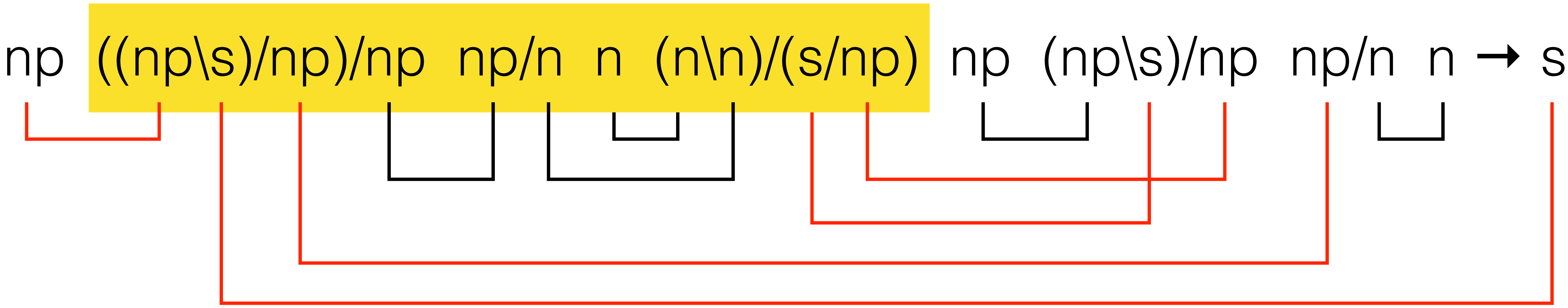
np/n n (n\n)/(s/np) np (np\s)/np (np\s)/np np ➝ s

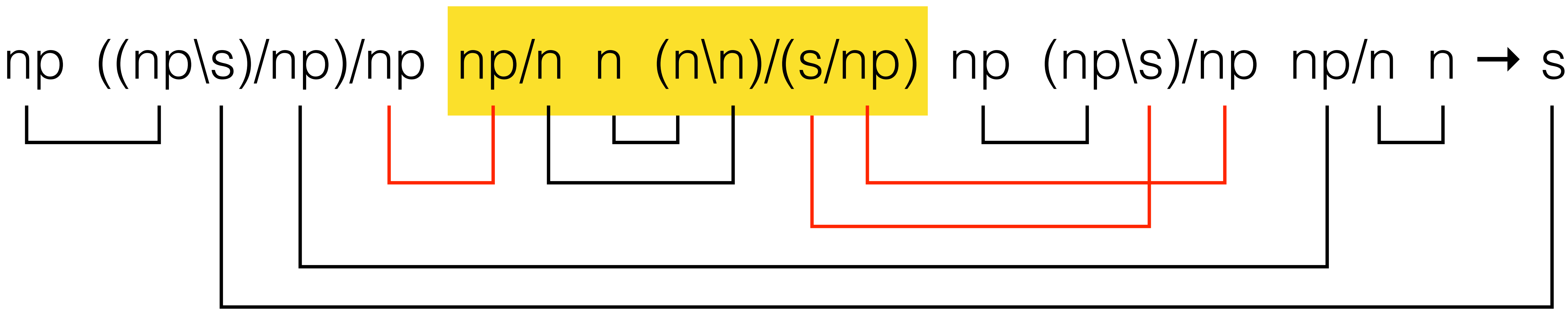
a woman who John saw saw John



Derivations with Cut only correspond to context-free derivations.

m = 4





One can always find a suitable subsequence of the antecedent for which the interpolant is "small".
Combinatorial property stemming from linearity and non-commutativity.

Lambek grammar G with semantics

m = maximal size of types used in G

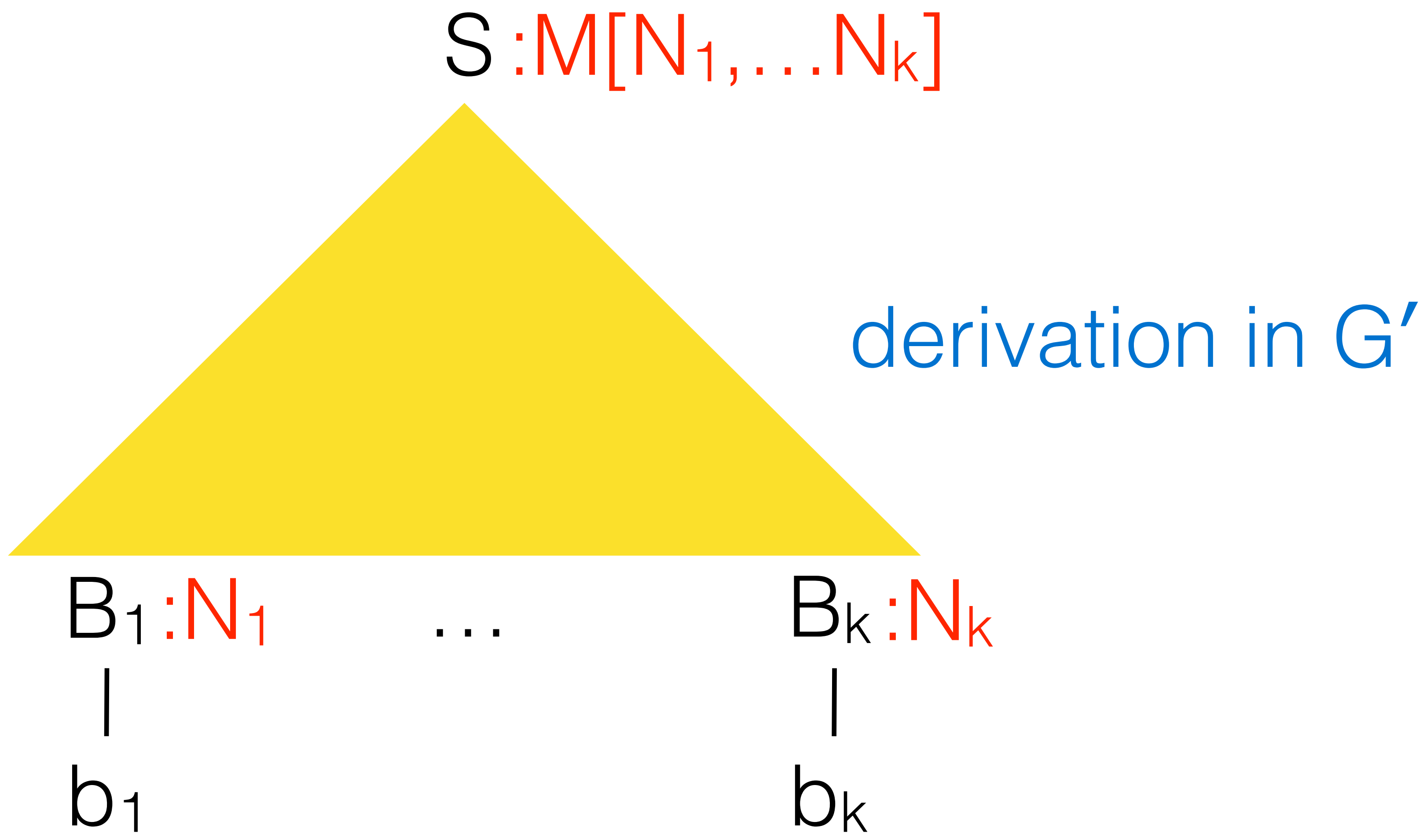
$$S_G = \{\, A_1 \ldots A_n \to A_{n+1} \mid \; \vdash A_1 \ldots A_n \to A_{n+1}, \; \mathrm{size}(A_i) \leq m \; (1 \leq i \leq n+1), \; \mathrm{size}(A_1) + \cdots + \mathrm{size}(A_n) \leq 2m \,\}$$

CFG G′

$A_{n+1} \to A_1 \ldots A_n : M$ $\quad$ ($A_1 \ldots A_n \to A_{n+1} \in S_G$)
$\quad \vdash A_1 \ldots A_n \to M{:}A_{n+1}$

$A \to a : M$ $\quad$ ($a{:}A{:}M$ is in G)

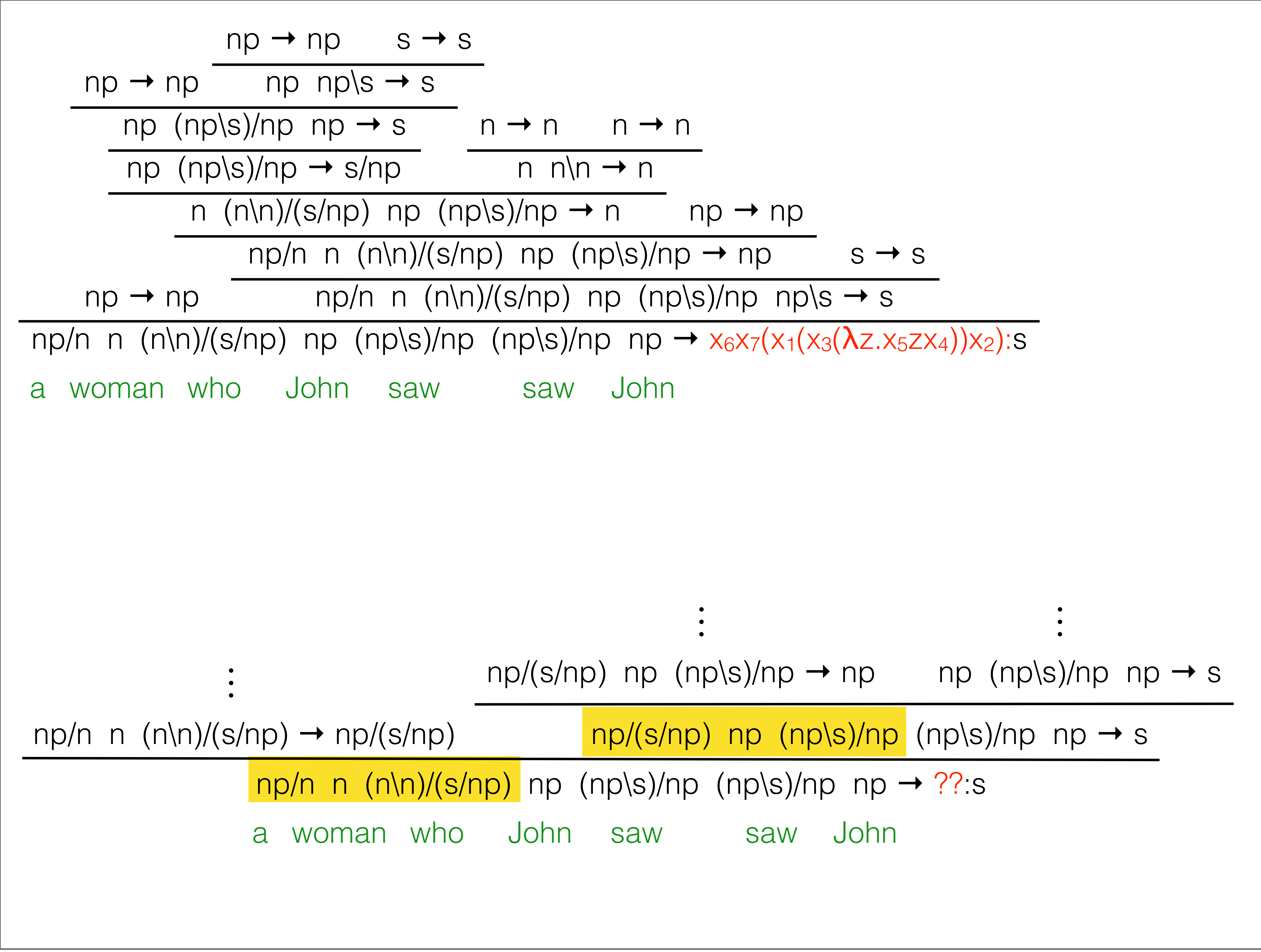
Use the same construction in the presence of semantics.
Linearity ensures that each derivable sequent has only finitely many cut-free derivations.



$$S_G \vdash_{Cut} B_1 \ldots B_k \rightarrow M[x_1,\ldots x_k]: S \qquad \frac{\Pi \rightarrow C \quad \Gamma\, C\, \Delta \rightarrow A}{\Gamma\, \Pi\, \Delta \rightarrow A}\,\text{Cut}$$

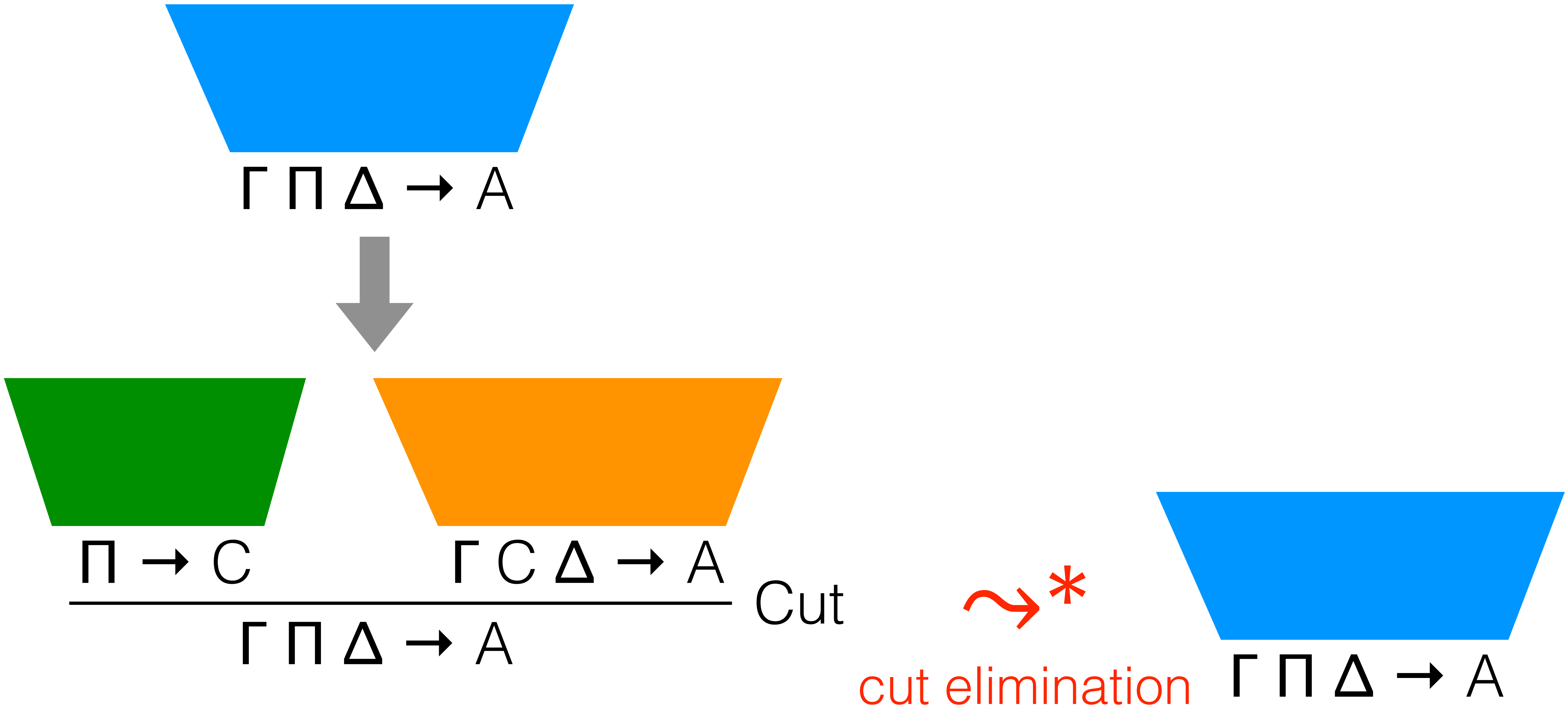
$$\vdash B_1 \ldots B_k \rightarrow M[x_1,\ldots x_k]:S \qquad R(G') \subseteq R(G) \;\checkmark$$

The inclusion in one direction continues to be easy.

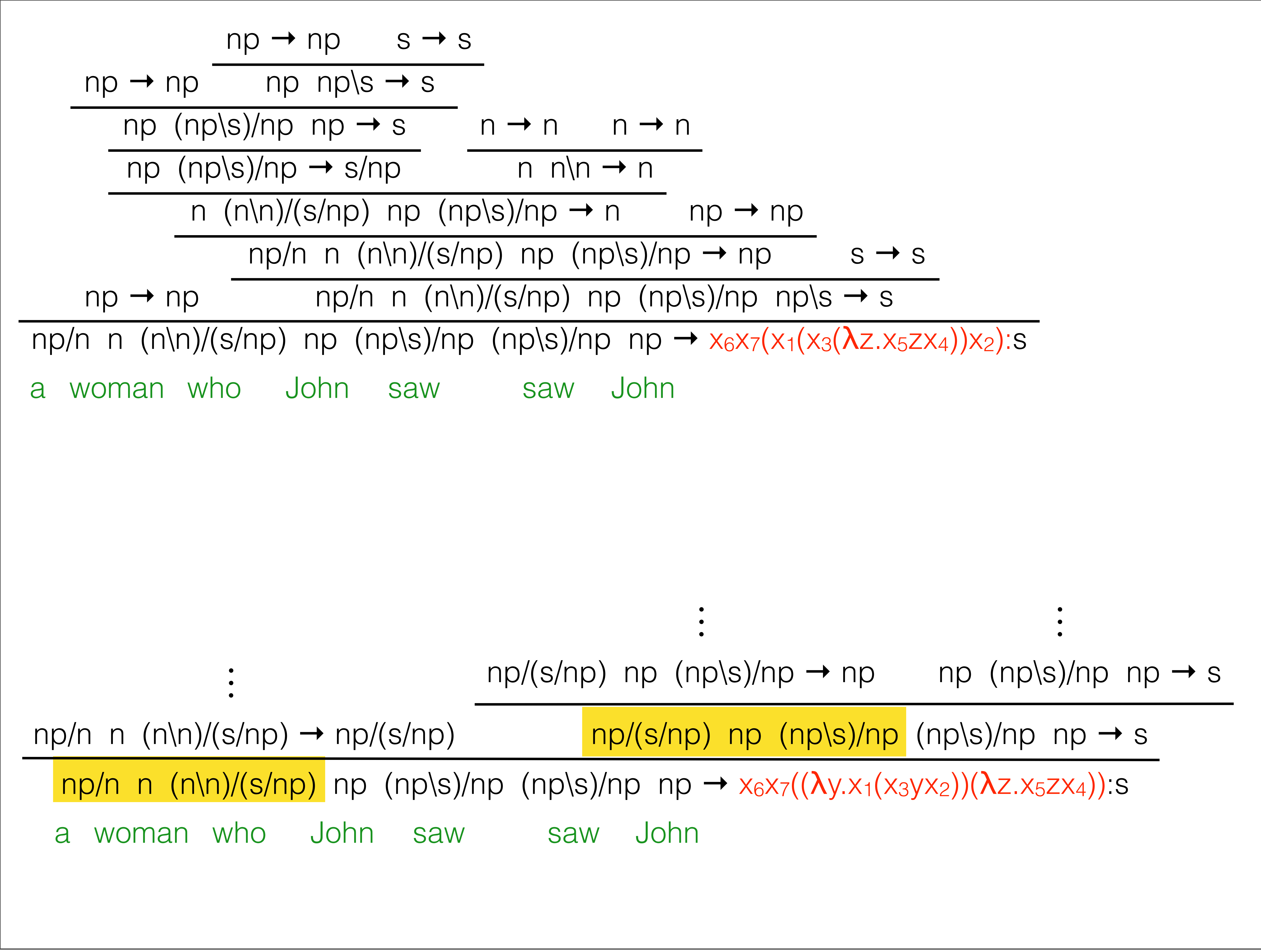


The other derivation depends on what derivation one gets by the interpolation algorithm.

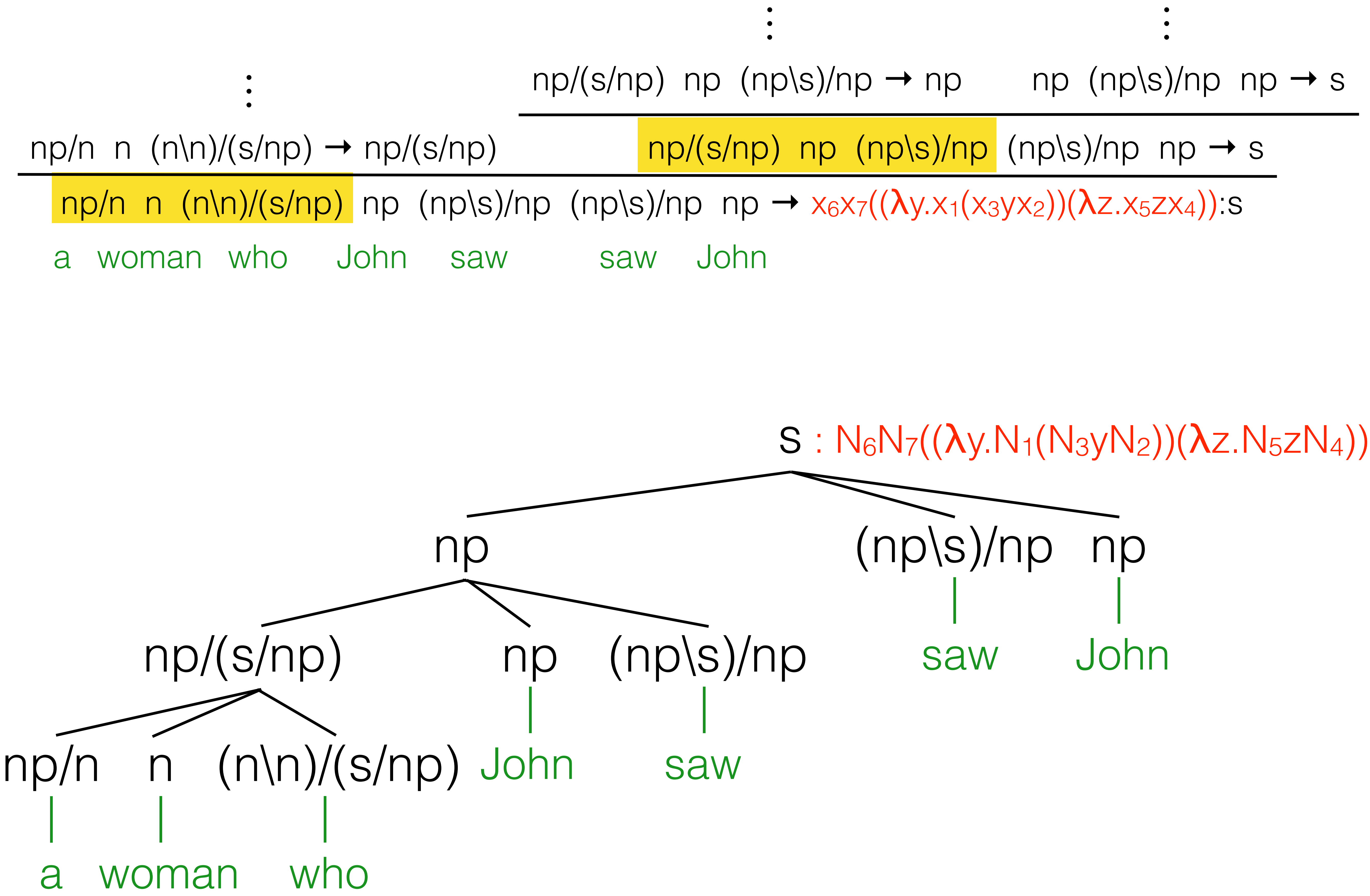
Interpolation



The syntactical proof of interpolation gives an algorithm that converts a cut-free derivation into two derivations that, when combined with Cut, reduce to the given derivation.



The non-normal lambda-term associated with the derivation with Cuts beta-reduces to the one associated with the original derivation.



This shows that the corresponding CFG derivation tree gives the right lambda-term.

Lambek vs. CFG

- Semantics-preserving left-corner transform
- Interpolation is reversed by cut elimination

The Lambek-to-CFG direction was brought up by Ian Pratt-Hartman in January 2007.