Cobordisms and commutative categorial grammars

Sergey Slavnov

National Research University Higher School of Economics
sslavnov@yandex.ru

Lambek grammars

- **Types:** $Tp ::= Prop | Tp \multimap Tp | Tp \circ\!- Tp$.
- **Rules:**
  - Elimination: $\dfrac{s : A \quad t : A \multimap B}{st : B}(\multimap E)$, $\dfrac{t : B \circ\!- A, \quad s : A}{ts : B}(\circ\!- E)$,
  - Introduction: $\dfrac{\dfrac{\dfrac{[x : A]}{\dots}}{xs : B}}{s : A \multimap B}(\multimap I)$, $\dfrac{\dfrac{\dfrac{[x : A]}{\dots}}{sx : B}}{s : B \circ\!- A}(\circ\!- I)$.
- **Example:**

$$\dfrac{\vdash \text{John} : NP \qquad \dfrac{\vdash \text{loves} : (NP \multimap S) \circ\!- NP \qquad \vdash \text{Mary} : NP}{\text{loves Mary} : NP \multimap S}}{\vdash \text{John loves Mary} : S}.$$

- **What else?**

Example: relativization

- **Add the axiom:**
  whom : $(NP \multimap NP) \circ\!\!- (S \circ\!\!- NP)$.
  - $$\frac{\vdash \text{John} : NP \quad \vdash \text{loves} : (NP \multimap S) \circ\!\!- NP \quad \vdash x : NP}{\vdash \text{John loves } x : S};$$
  - $\vdash$ John loves : $S \circ\!\!- NP$;
  - $\vdash$ whom John loves : $NP \multimap NP$;
  - $\vdash$ Mary whom John loves : NP.

Example: relativization

- **Add the axiom:**
  whom : $(NP \multimap NP) \circ\!- (S \circ\!- NP)$.
  - $$\frac{\vdash \text{John} : NP \quad \vdash \text{loves} : (NP \multimap S) \circ\!- NP \quad \vdash x : NP}{\vdash \text{John loves } x : S};$$
  - $\vdash$ John loves : $S \circ\!- NP$;
  - $\vdash$ whom John loves : $NP \multimap NP$;
  - $\vdash$ Mary whom John loves : NP.
- **But** add also:
  madly : $(NP \multimap S) \multimap (NP \multimap S)$.

Example: relativization

- **Add the axiom:**
  whom : $(NP \multimap NP) \multimapinv (S \multimapinv NP)$.
  - $$\frac{\vdash \text{John} : NP \quad \vdash \text{loves} : (NP \multimap S) \multimapinv NP \quad \vdash x : NP}{\vdash \text{John loves } x : S};$$
  - $\vdash$ John loves : $S \multimapinv NP$;
  - $\vdash$ whom John loves : $NP \multimap NP$;
  - $\vdash$ Mary whom John loves : NP.
- **But** add also:
  madly : $(NP \multimap S) \multimap (NP \multimap S)$.
  - $$\frac{\vdash x : NP \quad \cdots}{\vdash \text{John loves } x \text{ madly} : S};$$
  - $\cdots$

Example: relativization

- **Add the axiom:**
  whom : $(NP \multimap NP) \circ\!\!- (S \circ\!\!- NP)$.
  - $$\frac{\vdash \text{John} : NP \quad \vdash \text{loves} : (NP \multimap S) \circ\!\!- NP \quad \vdash x : NP}{\vdash \text{John loves } x : S};$$
  - $\vdash$ John loves : $S \circ\!\!- NP$;
  - $\vdash$ whom John loves : $NP \multimap NP$;
  - $\vdash$ Mary whom John loves : NP.
- **But** add also:
  madly : $(NP \multimap S) \multimap (NP \multimap S)$.
  - $$\frac{\vdash x : NP \quad \cdots}{\vdash \text{John loves } x \text{ madly} : S};$$
  - $\cdots$

- **Add the axiom:**
  whom : $(NP \multimap NP) \circ\!\!- (S \circ\!\!- NP)$.
  - $$\frac{\vdash \text{John} : NP \quad \vdash \text{loves} : (NP \multimap S) \circ\!\!- NP \quad \vdash x : NP}{\vdash \text{John loves } x : S};$$
  - $\vdash$ John loves : $S \circ\!\!- NP$;
  - $\vdash$ whom John loves : $NP \multimap NP$;
  - $\vdash$ Mary whom John loves : NP.
- **But** add also:
  madly : $(NP \multimap S) \multimap (NP \multimap S)$.
  - $$\frac{\vdash x : NP \quad \cdots}{\vdash \text{ John loves } x \text{ madly} : S};$$
  - $\cdots$

Example: relativization

- **Add the axiom:**
  whom : $(NP \multimap NP) \circ\!\!- (S \circ\!\!- NP)$.
  - $$\frac{\vdash \text{John} : NP \quad \vdash \text{loves} : (NP \multimap S) \circ\!\!- NP \quad \vdash x : NP}{\vdash \text{John loves } x : S};$$
  - $\vdash$ John loves : $S \circ\!\!- NP$;
  - $\vdash$ whom John loves : $NP \multimap NP$;
  - $\vdash$ Mary whom John loves : NP.
- **But** add also:
  madly : $(NP \multimap S) \multimap (NP \multimap S)$.
  - $$\frac{\vdash x : NP \quad \cdots}{\vdash \text{ John loves } x \text{ madly} : S};$$
  - That's it... :(
  - “Mary whom John loves madly” **underivable.**

Example: relativization

- **Add the axiom:**
  whom : $(NP \multimap NP) \circ\!\!- (S \circ\!\!- NP)$.
  - $$\frac{\vdash \text{John} : NP \quad \vdash \text{loves} : (NP \multimap S) \circ\!\!- NP \quad \vdash x : NP}{\vdash \text{John loves } x : S};$$
  - $\vdash$ John loves : $S \circ\!\!- NP$;
  - $\vdash$ whom John loves : $NP \multimap NP$;
  - $\vdash$ Mary whom John loves : NP.
- **But** add also:
  madly : $(NP \multimap S) \multimap (NP \multimap S)$.
  - $$\frac{\vdash x : NP \quad \cdots}{\vdash \text{ John loves } x \text{ madly} : S};$$
  - That's it... :(
  - "Mary whom John loves madly" **underivable.**
- Why not use commutative logic?

Abstract categorial grammars

- **Abstract categorial grammar (ACG):**
  Use implicational linear logic for types, encode everything in (linear) λ-terms, then translate to strings.
- **High level representation (abstract signature)**:
  - **Atomic types:** NP, S, $VP = NP \multimap S$.
  - **Axioms:**
    $\vdash JOHN : NP$, $\quad \vdash: MARY : NP$, $\quad \vdash LOVES : NP \multimap VP$,
    $\vdash MADLY : VP \multimap VP$, $\quad \vdash WHOM : (NP \multimap S) \multimap NP \multimap NP$.
- **Then**
  "Mary whom John loves madly"
  corresponds to
  $\vdash WHOM(\lambda x.(MADLY(LOVES \cdot x) \cdot JOHN)) : NP$.
- But how to obtain strings (*surface form*)?

ACG over string signature

- **String signature** over alphabet T:
  - **One atomic type:** O.
  - **Axioms:** $\vdash c : O \multimap O \ \forall c \in T$.
    *$O \multimap O$ is the type for strings.
  - **A word** $a_1 \dots a_n$ over T is represented as $a_1 \circ \dots \circ a_n$, where $M_1 \circ \dots \circ M_n = (\lambda t.M_1(\dots(M_n(t))\dots))$.
- **Lexicon** for translation:
  a homomorphism ϕ : {abstract signature} $\to$ {string signature}.
  - $\phi(A \multimap B) = \phi(A) \multimap \phi(B)$,
  - $\phi(x) = x$, for x a term variable,
    $\phi(M \cdot N) = \phi(M) \cdot \phi(N))$, $\phi(\lambda x.M) = (\lambda x.\phi(M))$,
  - for any abstract axiom $\vdash c : A$ holds
    $\vdash \phi(c) : \phi(A)$ in the string signature.
  - for the sentence type $\phi(S) = O \multimap O$

Example

- **Abstract signature:**
  $\vdash JOHN : NP$, $\vdash: MARY : NP$, $\vdash LOVES : NP \multimap VP$,
  $\vdash MADLY : VP \multimap VP$, $\vdash WHOM : (NP \multimap S) \multimap NP \multimap NP$
  $^{*}VP = NP \multimap S$.
- **Lexicon:**
  $\phi(NP) = \phi(S) = O \multimap O$,
  $\phi(JOHN) = \mathrm{John}$, $\phi(MARY) = \mathrm{Mary}$, $\phi(JIM) = \mathrm{Jim}$,
  $\phi(LOVES) = \lambda xy.(y \circ \mathrm{loves} \circ x)$, $\phi(MADLY) = \lambda fx.((f \cdot x) \circ (\mathrm{madly}))$,
  $\phi(WHOM) = \lambda fx.(x \circ (\mathrm{whom}) \circ (f \cdot (\lambda y.y)))$.
- **High level representation seems user friendly...**
  How about low level?
  Would like a more transparent surface representation

Boundaries

Definition

A *boundary* X consists of

- a natural number $|X|$, the *cardinality* of X,
- a subset $X_l \subseteq \{1, \ldots, |X|\}$, the *left boundary* of X.

- The *right boundary* X_r of X: $X_r = \{1, \ldots, |X|\} \setminus X_l$.
- Elements of X_l are *left endpoints* of X and have *left polarity*. Elements of X_r are *right endpoints* of X and have *right polarity*.

Boundaries

Definition

A *boundary* X consists of

- a natural number $|X|$, the *cardinality* of X,
- a subset $X_l \subseteq \{1, \ldots, |X|\}$, the *left boundary* of X.

I depict it as follows.

- **Example:** $|X| = 5$, $X_l = \{2, 5\}$.

$$X : \odot \quad \bullet \quad \odot \quad \odot \quad \bullet$$

Operations:

- **Tensor product:** disjoint union (concatenation).

  $X : \odot \ \bullet \ \odot \ \odot \ \bullet \qquad Y : \bullet \ \bullet \ \odot \ \bullet \ \odot$

  $X \otimes Y : \ \odot \ \bullet \ \odot \ \odot \ \bullet \ \bullet \ \bullet \ \odot \ \bullet \ \odot$

  $|X \otimes Y| = |X| + |Y|$, $(X \otimes Y)_l = X_l \cup (|X| + Y_l)$.
- **Duality:** order and polarity reversal.

  $X : \odot \ \bullet \ \odot \ \odot \ \bullet$

  $X : \odot \ \bullet \ \odot \ \odot \ \bullet \qquad X^{\perp} : \odot \ \bullet \ \bullet \ \odot \ \bullet$

  $|X^{\perp}| = |X|$, $(X^{\perp})_l = |X| + 1 - X_r$.
- Note: $(X \otimes Y)^{\perp} = Y^{\perp} \otimes X^{\perp}$.
- **Unit:** the empty boundary $\mathbf{1}$: $|\mathbf{1}| = 0$, $\mathbf{1}_l = \emptyset$.

Multiwords

Definition

A *regular multiword* with boundary X over an alphabet T is a directed graph with

- vertices enumerated from 1 to $|X|$,
- every edge (i, j) starting at a left endpoint $i \in X_l$ and ending at a right endpoint $j \in X_r$,
- every vertex adjacent to exactly one edge,
- edges labelled with words in T^*.

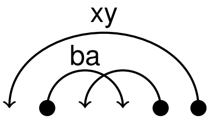
Multiwords

Definition

A *regular multiword* with boundary X over an alphabet T is a directed graph with

- vertices enumerated from 1 to $|X|$,
- every edge (i, j) starting at a left endpoint $i \in X_l$ and ending at a right endpoint $j \in X_r$,
- every vertex adjacent to exactly one edge,
- edges labelled with words in T^*.

Picture:

boundary: ⊙ ● ⊙ ⊙ ● ●   multiword: 

edges: $(6, xy, 1)$, $(2, ba, 4)$, $(5, , 3)$.

Multiwords

Definition

A *multiword* M with boundary X over an alphabet T is a pair $M = (M_0, M_c)$, where

- M_0, the *regular* part, is a regular multiword over T with the boundary X,
- M_c, the *singular* part, is a finite multiset of cyclic words over T.

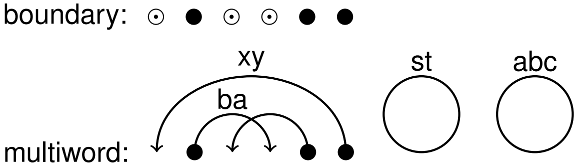
Multiwords

Definition

A *multiword* M with boundary X over an alphabet T is a pair $M = (M_0, M_c)$, where

- M_0, the *regular* part, is a regular multiword over T with the boundary X,
- M_c, the *singular* part, is a finite multiset of cyclic words over T.

Picture:



$M_0 = \{(6, xy, 1), (2, ba, 4), (5, , 3)\}$, $M_c = \{[st], [abc]\}$

Multiwords

Definition

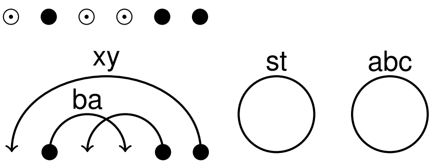
A *multiword* M with boundary X over an alphabet T is a pair $M = (M_0, M_c)$, where

- M_0, the *regular* part, is a regular multiword over T with the boundary X,
- M_c, the *singular* part, is a finite multiset of cyclic words over T.

Picture:

boundary:

multiword:



$M_0 = \{(6, xy, 1), (2, ba, 4), (5, , 3)\}$, $M_c = \{[st], [abc]\} \;\; = \{[ts], [bca]\}$.

Gluing

Edges in a multiword can be glued along vertices of opposite polarity.

- **Case 1:**

- **Case 2:**

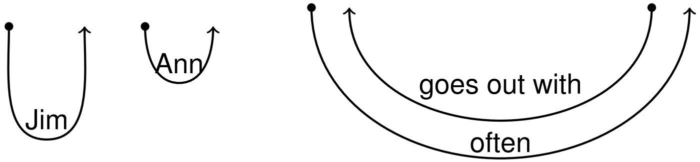
- **Case 3:**

Gluing

Multiple gluing:

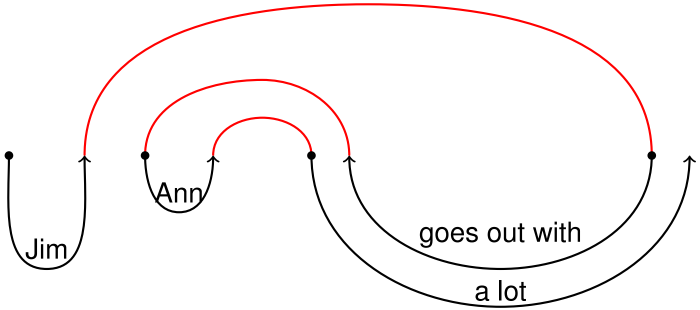
Gluing

Multiple gluing:



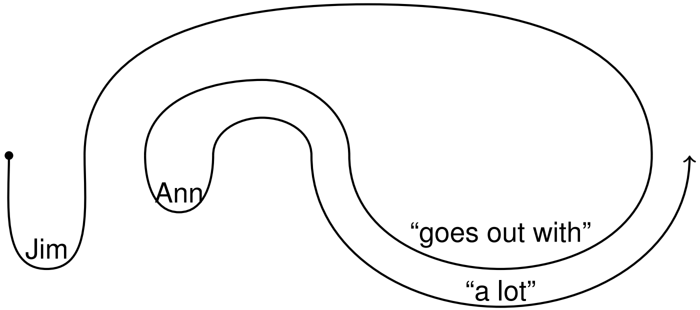
Gluing

Multiple gluing:

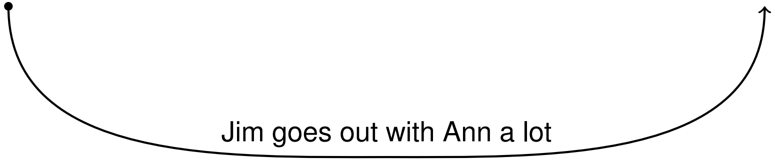


Gluing

Multiple gluing:



Multiple gluing:

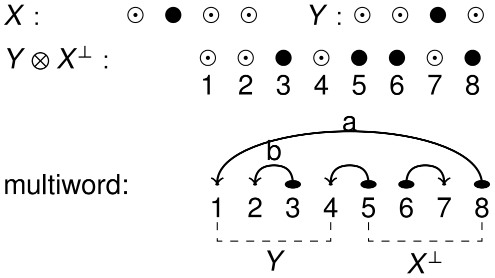


Word cobordisms

Definition

A *word cobordism* or *cowordism* $\sigma : X \to Y$ between boundaries X and Y over an alphabet T is a multiword over T with boundary $Y \otimes X^{\perp}$.

Picture:



Definition

A *word cobordism* or *cowordism* $\sigma : X \to Y$ between boundaries X and Y over an alphabet T is a multiword over T with boundary $Y \otimes X^{\perp}$.

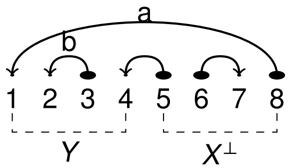
Picture:

X : ⊙ ● ⊙ ⊙ Y : ⊙ ⊙ ● ⊙

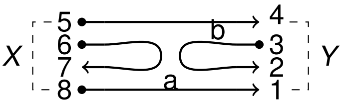
$Y \otimes X^{\perp}$: ⊙ ⊙ ● ⊙ ● ● ⊙ ●

1 2 3 4 5 6 7 8

multiword:



cowordsim:



Word cobordisms

Definition

A *word cobordism* or *cowordism* $\sigma : X \to Y$ between boundaries X and Y over an alphabet T is a multiword over T with boundary $Y \otimes X^{\perp}$.

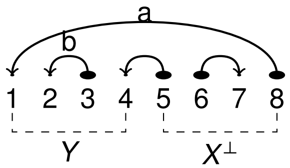
Picture:

X : ⊙ ● ⊙ ⊙ Y : ⊙ ⊙ ● ⊙

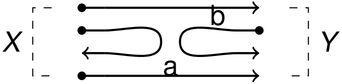
$Y \otimes X^{\perp}$: ⊙ ⊙ ● ⊙ ● ● ⊙ ●

1 2 3 4 5 6 7 8

multiword:



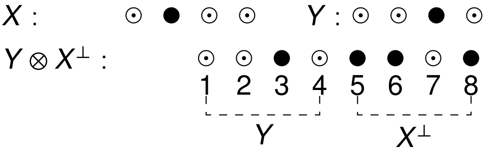
cowordsim:



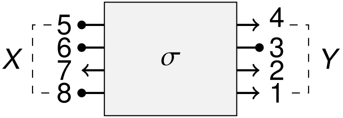
Definition

A *word cobordism* or *cowordism* $\sigma : X \to Y$ between boundaries X and Y over an alphabet T is a multiword over T with boundary $Y \otimes X^{\perp}$.

Picture:



$\sigma : X \to Y$



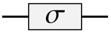
Word cobordisms

Definition

A *word cobordism* or *cowordism* $\sigma : X \to Y$ between boundaries X and Y over an alphabet T is a multiword over T with boundary $Y \otimes X^{\perp}$.

Squeezed picture:

- For $\sigma : X \to Y$

  $\sigma : X$ 

 Y

Word cobordisms

Definition

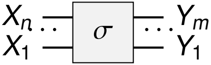
A *word cobordism* or *cowordism* $\sigma : X \to Y$ between boundaries X and Y over an alphabet T is a multiword over T with boundary $Y \otimes X^{\perp}$.

Squeezed picture:

- For $\sigma : X \to Y$

  $\sigma :$ 

- For $\sigma : X_1 \otimes \ldots \otimes X_n \to Y_1 \otimes \ldots \otimes Y_m$

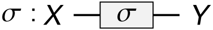
  $\sigma :$ 

Definition

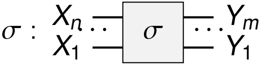
A *word cobordism* or *cowordism* $\sigma : X \to Y$ between boundaries X and Y over an alphabet T is a multiword over T with boundary $Y \otimes X^\perp$.

Squeezed picture:

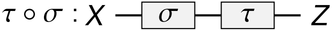
- For $\sigma : X \to Y$



- For $\sigma : X_1 \otimes \ldots \otimes X_n \to Y_1 \otimes \ldots \otimes Y_m$



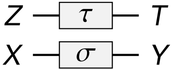
- **Composition:**



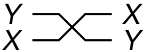
Category

The category $\mathbf{Coword_T}$ of cowordisms over an alphabet T:

- **Identities:** $\mathrm{id}_X : X$ — X .
- **Tensor product:**

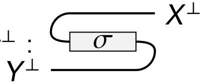
  $\sigma \otimes \tau$:
  

- **Symmetries:**

  s_{XY} :
  

- **Duality:**

  σ :
  

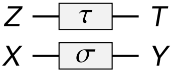
  $\sigma^\perp$:
  

Category

The category $\mathbf{Coword_T}$ of cowordisms over an alphabet T:

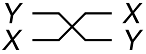
- **Identities:** $\mathrm{id}_X : X$ — X .
- **Tensor product:**

$\sigma \otimes \tau :$

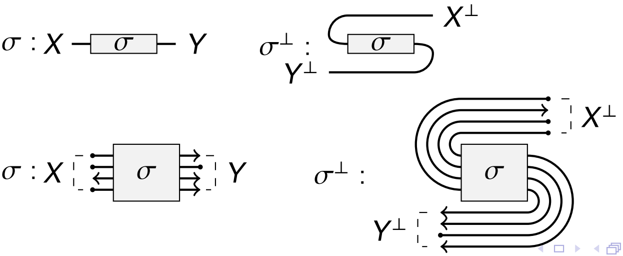


- **Symmetries:**

$s_{XY} :$

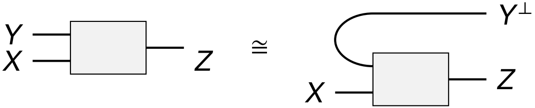


- **Duality:**



Compact structure

- $Hom(X \otimes Y, Z) = Hom(X, (Y \otimes Z^{\perp})^{\perp}) = Hom(X, Z \otimes Y^{\perp})$

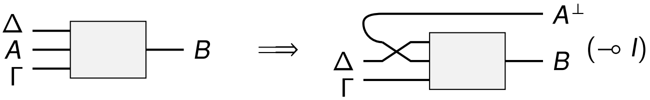


The category $\mathbf{Coword_T}$ is compact.

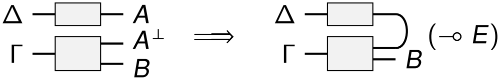
- In a sense, which can be made precise, the category $\mathbf{Coword_T}$ is a *free compact category* generated by the free monoid T^*, where T^* is considered as a category with single object.
- As any compact category, $\mathbf{Coword_T}$ is a model of *multiplicative linear logic* (**MLL**), *multiplicative intuitionistic linear logic* (**MILL**) and *linear λ-calculus*.

Interpretation of linear λ-calculus

- **Interpretation of types:**
  - Assign a boundary to each atomic type $p \mapsto [[p]]$.
  - $[[A \multimap B]] = [[B]] \otimes [[A]]^{\perp}$.
- A derivable typing judgement $x_1 : A_1, \ldots, x_n; A_m \vdash t : A$ is interpreted as a cowordism $[[A_1]] \otimes \ldots \otimes [[A_n]] \to [[A]]$.
- **Rules:**
  - $(\multimap \mathrm{I})$: $\dfrac{\Gamma, x{:}A, \Delta \vdash t{:}B}{\Gamma, \Delta \vdash (\lambda x.t){:}A \multimap B}$



  - $(\multimap \mathrm{E})$: $\dfrac{\Delta \vdash s{:}A, \qquad \Gamma \vdash t{:}A \multimap B}{\Gamma, \Delta \vdash (ts){:}B}$



Representing string ACG

- **String ACG**
  - abstract signature;
  - string signature over T:
    - string type $O \multimap O$,
    - axioms $\vdash c : O \multimap O \; \forall c \in T$;
  - homomorphism ϕ : {abstract signature} $\to$ {string signature}.
- **Cowordism representation** of the string signature:
  - Type $O \quad \Rightarrow \quad$ boundary $\quad \odot$;
  - axiom $c : O \multimap O \quad \Rightarrow \quad$ cowordism  .
- **Cowordism representation** of string ACG:

$$\{\text{abstract signature}\} \xrightarrow{\phi} \{\text{string signature}\} \longrightarrow \{\text{cowordisms}\}.$$

Example

- **Abstract signature:**

$$\vdash JOHN : NP, \vdash MARY : NP, \vdash LOVES : NP \multimap NP \multimap S,$$
$$\vdash MADLY : (NP \multimap S) \multimap NP \multimap S,$$
$$\vdash WHOM : (NP \multimap S) \multimap NP \multimap NP.$$

Example

- **Abstract signature:**

  $\vdash JOHN : NP, \vdash MARY : NP, \vdash LOVES : NP \multimap NP \multimap S,$
  $\vdash MADLY : (NP \multimap S) \multimap NP \multimap S,$
  $\vdash WHOM : (NP \multimap S) \multimap NP \multimap NP.$

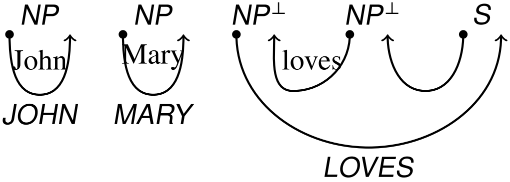
- **Cowordism representation:**

Example

- **Abstract signature:**

  $\vdash JOHN : NP, \vdash MARY : NP, \vdash LOVES : NP \multimap NP \multimap S,$
  $\vdash MADLY : (NP \multimap S) \multimap NP \multimap S,$
  $\vdash WHOM : (NP \multimap S) \multimap NP \multimap NP.$

- **Cowordism representation:**



Example

- **Abstract signature:**

$$\vdash JOHN : NP, \vdash MARY : NP, \vdash LOVES : NP \multimap NP \multimap S,$$
$$\vdash MADLY : (NP \multimap S) \multimap NP \multimap S,$$
$$\vdash WHOM : (NP \multimap S) \multimap NP \multimap NP.$$

- **Cowordism representation:**

Example

- **Abstract signature:**

$\vdash JOHN : NP, \vdash MARY : NP, \vdash LOVES : NP \multimap NP \multimap S,$
$\vdash MADLY : (NP \multimap S) \multimap NP \multimap S,$
$\vdash WHOM : (NP \multimap S) \multimap NP \multimap NP.$

- **Cowordism representation:**

Example

- **Abstract signature:**

  $\vdash JOHN : NP, \vdash MARY : NP, \vdash LOVES : NP \multimap NP \multimap S,$
  $\vdash MADLY : (NP \multimap S) \multimap NP \multimap S,$
  $\vdash WHOM : (NP \multimap S) \multimap NP \multimap NP.$
- **Cowordism representation:**
  $([[(NP \multimap S) \multimap NP \multimap S]] = [[S]] \otimes [[NP^{\perp}]] \otimes [[NP]] \otimes [[S^{\perp}]].)$

Example

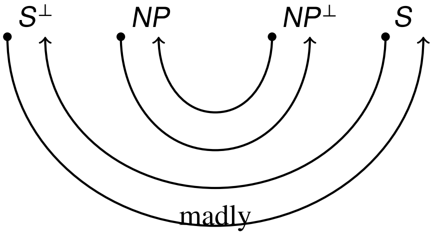
- **Abstract signature:**

$\vdash JOHN : NP, \vdash MARY : NP, \vdash LOVES : NP \multimap NP \multimap S,$
$\vdash MADLY : (NP \multimap S) \multimap NP \multimap S,$
$\vdash WHOM : (NP \multimap S) \multimap NP \multimap NP.$

- **Cowordism representation:**

$([[(NP \multimap S) \multimap NP \multimap S]] = [[S^\perp]] \otimes [[NP]] \otimes [[NP^\perp]] \otimes [[S]].)$

$MADLY$:



Example

- **Abstract signature:**

$$\vdash JOHN : NP, \vdash MARY : NP, \vdash LOVES : NP \multimap NP \multimap S,$$
$$\vdash MADLY : (NP \multimap S) \multimap NP \multimap S,$$
$$\vdash WHOM : (NP \multimap S) \multimap NP \multimap NP.$$

- **Cowordism representation:**

Example

- **Abstract signature:**

$$\vdash JOHN : NP, \vdash MARY : NP, \vdash LOVES : NP \multimap NP \multimap S,$$
$$\vdash MADLY : (NP \multimap S) \multimap NP \multimap S,$$
$$\vdash WHOM : (NP \multimap S) \multimap NP \multimap NP.$$

- **Cowordism representation:**

Example

- **Abstract signature:**

  $\vdash JOHN : NP, \vdash MARY : NP, \vdash LOVES : NP \multimap NP \multimap S,$
  $\vdash MADLY : (NP \multimap S) \multimap NP \multimap S,$
  $\vdash WHOM : (NP \multimap S) \multimap NP \multimap NP.$
- **Cowordism representation:**
  $([[(NP \multimap S) \multimap NP \multimap NP]] = [[NP]] \otimes [[NP^{\perp}]] \otimes [[NP]] \otimes [[S^{\perp}]].)$

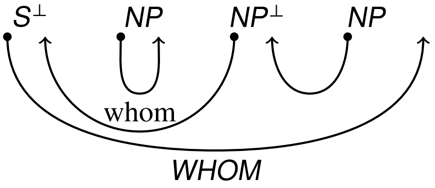
Example

- **Abstract signature:**

  $\vdash JOHN : NP, \vdash MARY : NP, \vdash LOVES : NP \multimap NP \multimap S,$
  $\vdash MADLY : (NP \multimap S) \multimap NP \multimap S,$
  $\vdash WHOM : (NP \multimap S) \multimap NP \multimap NP.$

- **Cowordism representation:**

  $([[(NP \multimap S) \multimap NP \multimap NP]] = [[S^{\perp}]] \otimes [[NP]] \otimes [[NP^{\perp}]] \otimes [[NP]].)$

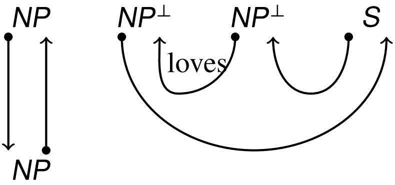


Derivation

$$\frac{x : NP \vdash x : NP \quad \vdash LOVES : NP \multimap NP \multimap S}{x : NP \vdash LOVES \cdot x : NP \multimap S}$$

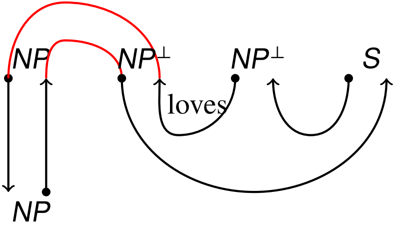
Derivation

$$\frac{x : NP \vdash x : NP \quad \vdash LOVES : NP \multimap NP \multimap S}{x : NP \vdash LOVES \cdot x : NP \multimap S}$$



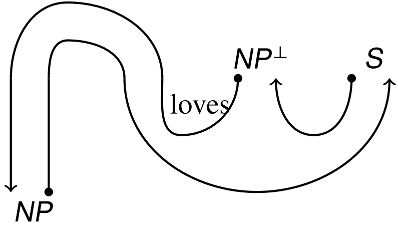
Derivation

$$\frac{x : NP \vdash x : NP \qquad \vdash LOVES : NP \multimap NP \multimap S}{x : NP \vdash LOVES \cdot x : NP \multimap S}$$



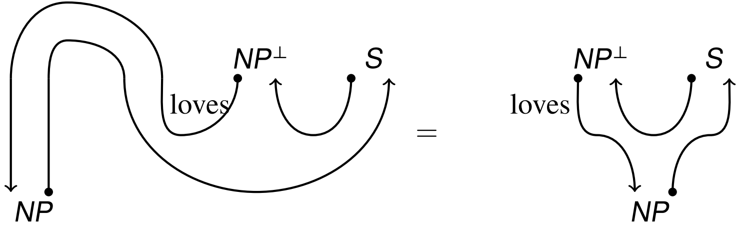
Derivation

$$\frac{x : NP \vdash x : NP \qquad \vdash LOVES : NP \multimap NP \multimap S}{x : NP \vdash LOVES \cdot x : NP \multimap S}$$



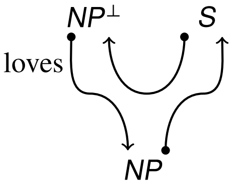
Derivation

$$\frac{x : NP \vdash x : NP \qquad \vdash LOVES : NP \multimap NP \multimap S}{x : NP \vdash LOVES \cdot x : NP \multimap S}$$



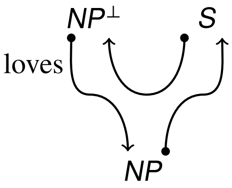
Derivation

$$\frac{x : NP \vdash x : NP \qquad \vdash LOVES : NP \multimap NP \multimap S}{x : NP \vdash LOVES \cdot x : NP \multimap S}$$



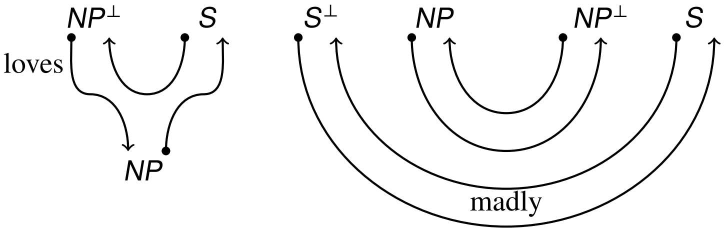
Derivation

$$x : NP \vdash LOVES \cdot x : NP \multimap S$$



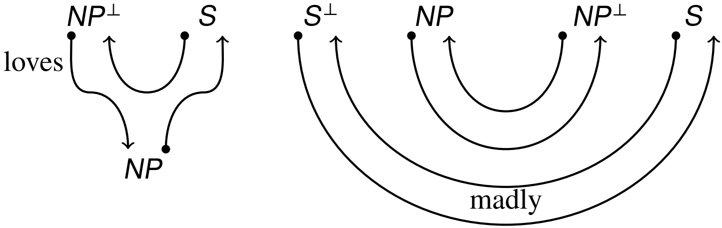
Derivation

$x : NP \vdash LOVES \cdot x : NP \multimap S \qquad \vdash MADLY : (NP \multimap S) \multimap NP \multimap S$



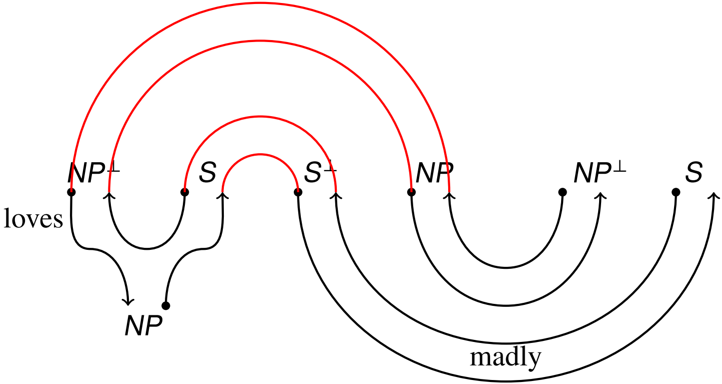
Derivation

$$\frac{x : NP \vdash LOVES \cdot x : NP \multimap S \qquad \vdash MADLY : (NP \multimap S) \multimap NP \multimap S}{x : NP \vdash MADLY(LOVES \cdot x) : NP \multimap S}$$



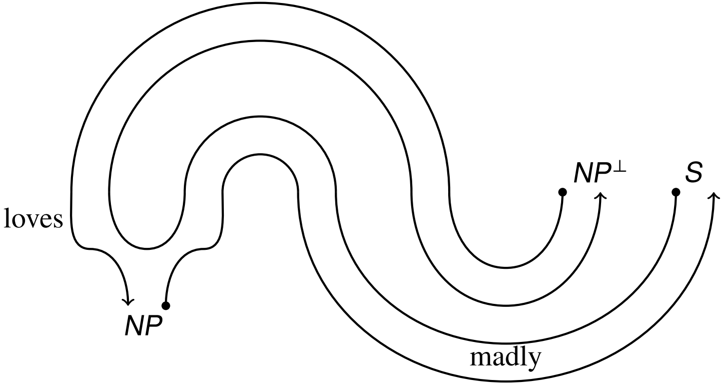
Derivation

$$\frac{x : NP \vdash LOVES \cdot x : NP \multimap S \qquad \vdash MADLY : (NP \multimap S) \multimap NP \multimap S}{x : NP \vdash MADLY(LOVES \cdot x) : NP \multimap S}$$



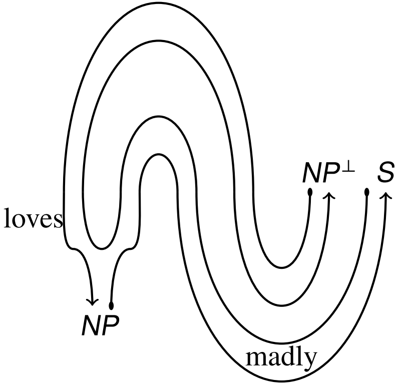
Derivation

$$\frac{x : NP \vdash LOVES \cdot x : NP \multimap S \qquad \vdash MADLY : (NP \multimap S) \multimap NP \multimap S}{x : NP \vdash MADLY(LOVES \cdot x) : NP \multimap S}$$



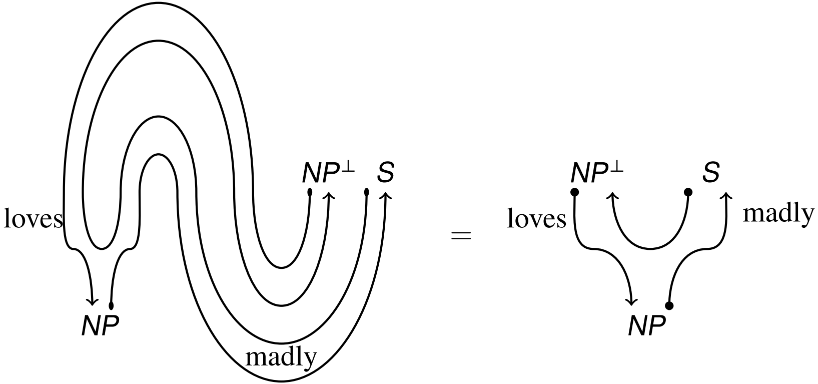
Derivation

$$\frac{x : NP \vdash LOVES \cdot x : NP \multimap S \qquad \vdash MADLY : (NP \multimap S) \multimap NP \multimap S}{x : NP \vdash MADLY(LOVES \cdot x) : NP \multimap S}$$



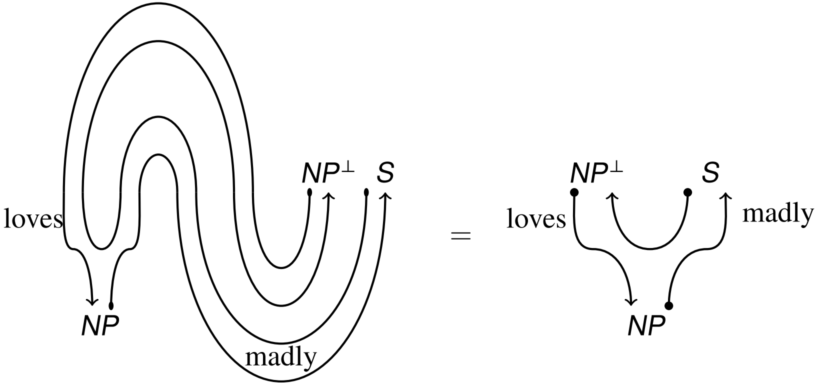
Derivation

$$\frac{x : NP \vdash LOVES \cdot x : NP \multimap S \qquad \vdash MADLY : (NP \multimap S) \multimap NP \multimap S}{x : NP \vdash MADLY(LOVES \cdot x) : NP \multimap S}$$



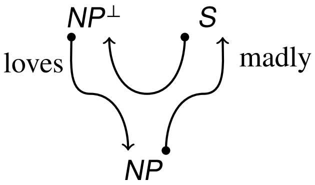
Derivation

$x : NP \vdash MADLY(LOVES \cdot x) : NP \multimap S$



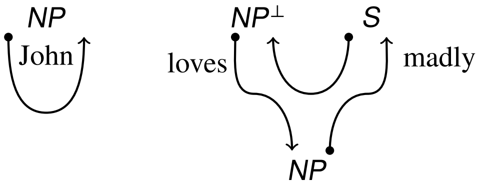
Derivation

$x : NP \vdash MADLY(LOVES \cdot x) : NP \multimap S$



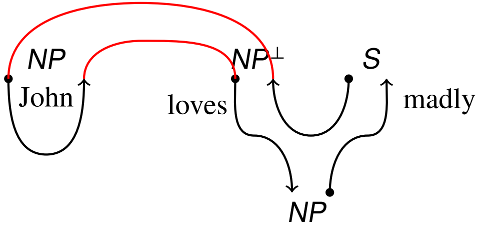
Derivation

$\vdash JOHN : NP \qquad x : NP \vdash MADLY(LOVES \cdot x) : NP \multimap S$



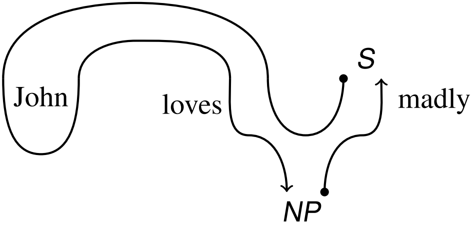
Derivation

$$\frac{\vdash JOHN : NP \qquad x : NP \vdash MADLY(LOVES \cdot x) : NP \multimap S}{x : NP \vdash MADLY(LOVES \cdot x)JOHN : S}$$



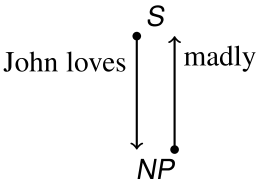
Derivation

$$\frac{\vdash JOHN : NP \qquad x : NP \vdash MADLY(LOVES \cdot x) : NP \multimap S}{x : NP \vdash MADLY(LOVES \cdot x)JOHN : S}$$



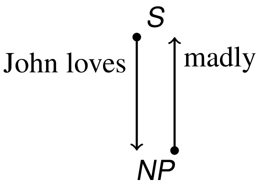
Derivation

$$\frac{\vdash JOHN : NP \qquad x : NP \vdash MADLY(LOVES \cdot x) : NP \multimap S}{x : NP \vdash MADLY(LOVES \cdot x)JOHN : S}$$



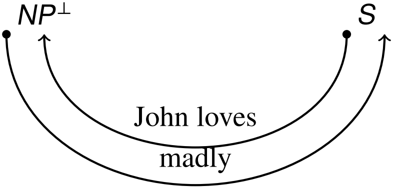
Derivation

$x : NP \vdash MADLY(LOVES \cdot x)JOHN : S$



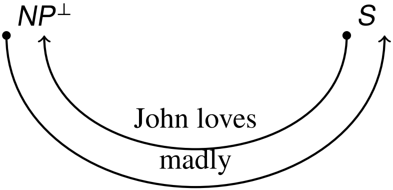
Derivation

$$\frac{x : NP \vdash MADLY(LOVES \cdot x)JOHN : S}{\vdash \lambda x.MADLY(LOVES \cdot x)JOHN : NP \multimap S}$$



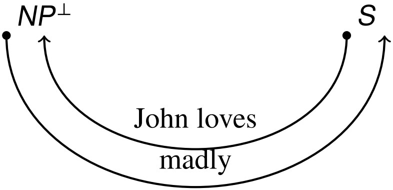
Derivation

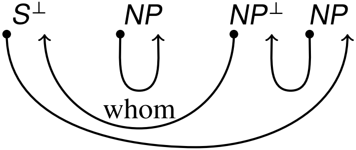
$\vdash \lambda x.MADLY(LOVES \cdot x)JOHN : NP \multimap S$



Derivation

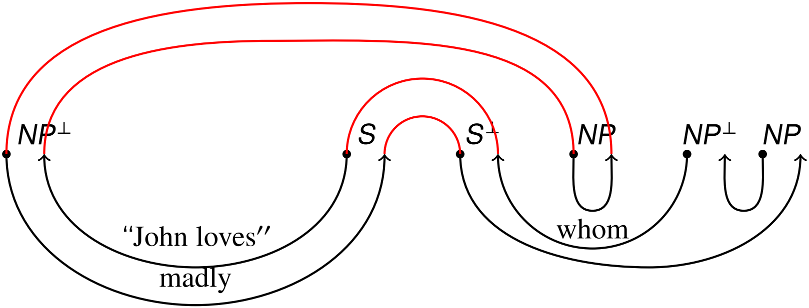
$\vdash \lambda x.MADLY(LOVES \cdot x)JOHN : NP \multimap S \qquad \vdash WHOM : (NP \multimap S) \multimap NP \multimap NP$





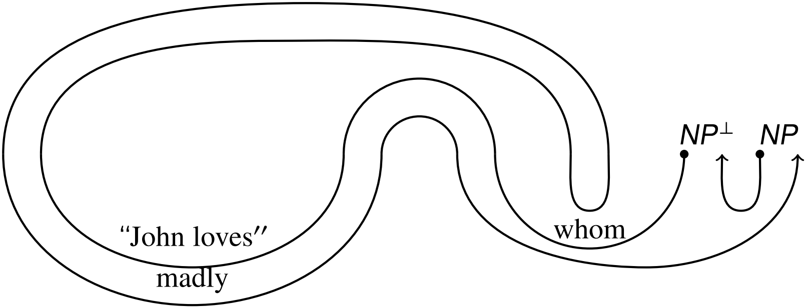
Derivation

$$\frac{\vdash \lambda x.MADLY(LOVES \cdot x)JOHN : NP \multimap S \qquad \vdash WHOM : (NP \multimap S) \multimap NP \multimap NP}{WHOM(\lambda x.MADLY(LOVES \cdot x)JOHN) : NP \multimap NP}$$



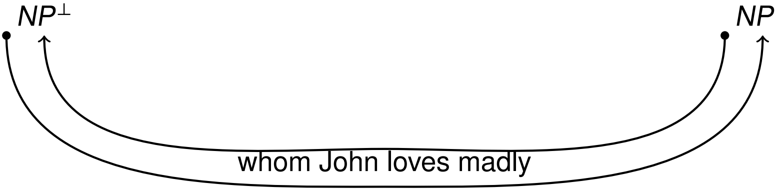
Derivation

$$\frac{\vdash \lambda x.MADLY(LOVES \cdot x)JOHN : NP \multimap S \qquad \vdash WHOM : (NP \multimap S) \multimap NP \multimap NP}{WHOM(\lambda x.MADLY(LOVES \cdot x)JOHN) : NP \multimap NP}$$



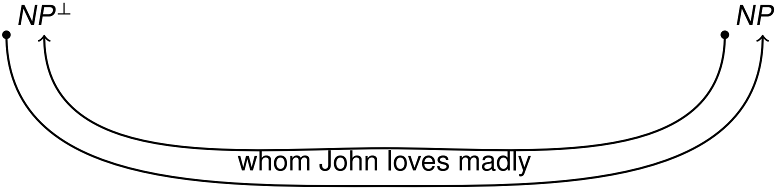
Derivation

$$\frac{\vdash \lambda x.MADLY(LOVES \cdot x)JOHN : NP \multimap S \qquad \vdash WHOM : (NP \multimap S) \multimap NP \multimap NP}{WHOM(\lambda x.MADLY(LOVES \cdot x)JOHN) : NP \multimap NP}$$



Derivation

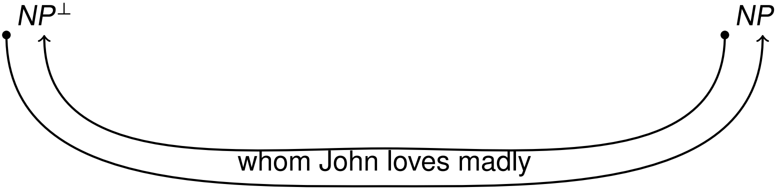
$$\frac{\vdash \lambda x.MADLY(LOVES \cdot x)JOHN : NP \multimap S \qquad \vdash WHOM : (NP \multimap S) \multimap NP \multimap NP}{WHOM(\lambda x.MADLY(LOVES \cdot x)JOHN) : NP \multimap NP}$$



Etc.

Derivation

$$\frac{\vdash \lambda x.MADLY(LOVES \cdot x)JOHN : NP \multimap S \qquad \vdash WHOM : (NP \multimap S) \multimap NP \multimap NP}{WHOM(\lambda x.MADLY(LOVES \cdot x)JOHN) : NP \multimap NP}$$



Etc.

One more step and we get *NP*

Mary whom John loves madly.

Digression: about meaning models

- **Montague-style semantics** (possible worlds etc):
  - higher order logic, λ-calculus;
  - socializes well with *intuitionistic* logic grammars (Lambek, ACG, ...).
- **Distributional models** (*real world*, close to applications):
  - meaning $\approx$ vector in the space spanned by context words,
    *say, the probability of co-occurrence in the corpus.
  - No metaphysics. No logic either.
- **DisCoCat** (*distributional compositional categorical semantics*):
  - considers vector spaces of distributional models as a *category*,
  - constructs a functor
    $\{$logic grammars$\} \to \{$vector spaces$\}$.
- The target category **FdVec** of **DisCoCat** semantics:
  - compact closed (like cowordisms),
  - "classical" rather than "intuitionistic" (involutive duality).
  - Cannot we try *classical* linear logic on the *syntactic* side as well?

Interpretation of MLL

- **Linear logic:**
  - Given a set N of *positive literals*.
  - The set $Fm(N)$ of formulas is defined by
    $Prop ::= N|N^{\perp}, \quad Fm ::= Prop|Fm \otimes Fm|Fm \wp Fm.$
  - Defined connectives:
    $(P^{\perp})^{\perp} = P$, for $P \in N$, $(A \otimes B)^{\perp} = A^{\perp} \wp B^{\perp}$, $(A \wp B)^{\perp} = A^{\perp} \otimes B^{\perp}$.
    $A \multimap B = A^{\perp} \wp B$.
- **Interpretation:**
  - assign a boundary $p \mapsto [[p]]$ for $p \in N$.
  - $[[A \otimes B]] = [[A]] \otimes [[B]]$, $[[A \wp B]] = [[B]] \otimes [[A]]$,
    $[[A^{\perp}]] = [[A]]^{\perp}$.
  - For a *sequent* $\Gamma = A_1, \ldots, A_n$
    $[[\Gamma]] = [[A_1 \wp \ldots \wp A_n]] = [[A_n]] \otimes \ldots \otimes [[A_1]]$.
- A proof σ of the sequent Γ will be interpreted as a cowordism $[[\sigma]] : \mathbf{1} \to [[\Gamma]]$.

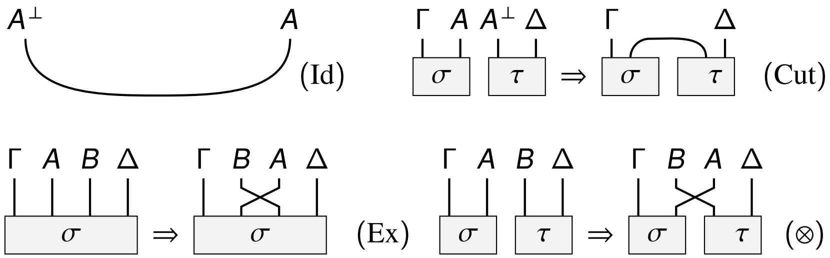
Interpretation of proofs

- **Rules of MLL:**

$$\vdash A^{\perp}, A \text{ (Id)}, \quad \frac{\vdash \Gamma, A \quad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta} \text{ (Cut)},$$

$$\frac{\vdash \Gamma, A, B, \Delta}{\vdash \Gamma, B, A, \Delta} \text{ (Ex)}, \quad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \wp B} (\wp), \quad \frac{\vdash \Gamma, A \quad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta} (\otimes).$$

- **Interpretation:** (pictures rotated 90° counterclockwise)



Let an interpretation of $Fm(N)$ in $\mathbf{Coword_T}$ be given.

- A *cowordism typing judgement* is an expression of the form

$$\frac{\sigma}{\vdash \Gamma},$$

where Γ is a sequent, and

$$\sigma : \mathbf{1} \to [[\Gamma]]$$

is a cowordism.

- A *cowordism signature* Σ is a set of cowordism typing judgements, called *axioms*.
- A cowordism typing judgement is *derivable from* Σ if it can be obtained from the axioms by rules of **MLL**.

Grammars

Definition

A *linear logic grammar* (**LLG**) or a *cobordism grammar*, or a *tensor grammar* G consists of

- a finite set N of positive literals;
- a finite alphabet T of *terminals*;
- an interpretation of $Fm(N)$ in $\mathbf{Coword_T}$;
- a cowordism signature Lex, the *lexicon*, with finitely many axioms;
- a *sentence type* $S \in N$, such that $[[S]]$ has exactly one left and exactly one right endpoint.

The *language generated by* G is the set of *regular* cowordisms σ for which

$$\frac{\sigma}{\vdash S}$$

is derivable from Lex.

Expressivity

- **Abstract categorial grammars:** String and tree ACG embed into LLG isomorphically and conservatively.
  Question: A language generated by a string ACG is generated by an LLG as well. Is the opposite true? (Probably yes.)
- **Multiple context-free grammars:** MCFG embed into LLG. A language is multiple context-free iff it is generated by an LLG with a $\otimes$-free lexicon.
  Corollary: Second order string ACG are multiple context-free. (But this is known.)
- **Complexity:** At least as bad as ACG. In the unlexicalized case even decidability is unknown. A general LLG can generate an NP-complete language.

Subset sum problem

Subset sum problem

Given a finite sequence s of integers, determine if there is a subsequence $s' \subseteq s$ such that $\sum\limits_{z \in s'} z = 0$. (NP-complete.)

Subset sum problem

Generating solutions to SSP.

Subset sum problem

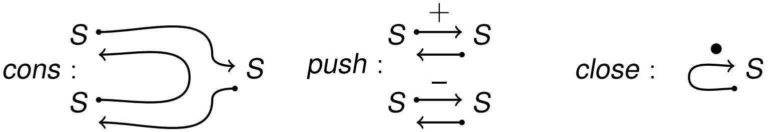
Generating solutions to SSP.

- We encode integers as words in the alphabet $\{+, -\}$, and integer lists as words in the alphabet $T = \{+, -, \bullet\}$, with $\bullet$ as a separation sign.

Subset sum problem

Generating solutions to SSP.

- We encode integers as words in the alphabet $\{+, -\}$, and integer lists as words in the alphabet $T = \{+, -, \bullet\}$, with $\bullet$ as a separation sign.
- Cowordisms for lists summing to zero:

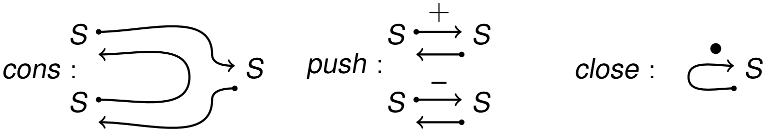


cons generates lists with arbitrary many empty slots.
push fills slots (always in pairs).
close closes slots.
Together with symmetries, these generate all lists summing to zero.

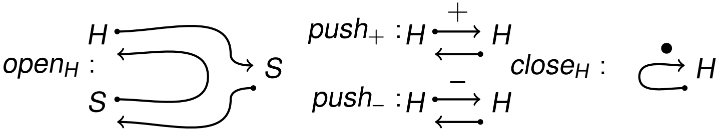
Subset sum problem

Generating solutions to SSP.

- We encode integers as words in the alphabet $\{+,-\}$, and integer lists as words in the alphabet $T = \{+,-,\bullet\}$, with $\bullet$ as a separation sign.
- Cowordisms for lists summing to zero:



- Cowordisms for lists with deceptive slots:

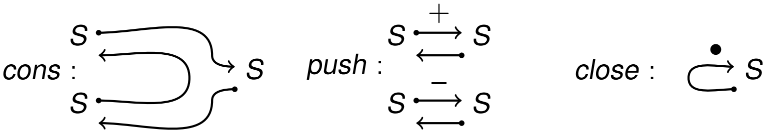


open$_H$ adds deceptive slots to the list, *push$_-$* and *push$_+$* fill deceptive slots, *close$_H$* closes deceptive slots. These generate all solutions to SSP.

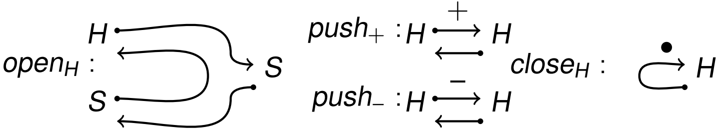
Subset sum problem

Generating solutions to SSP.

- We encode integers as words in the alphabet $\{+,-\}$, and integer lists as words in the alphabet $T = \{+,-,\bullet\}$, with $\bullet$ as a separation sign.
- Cowordisms for lists summing to zero:



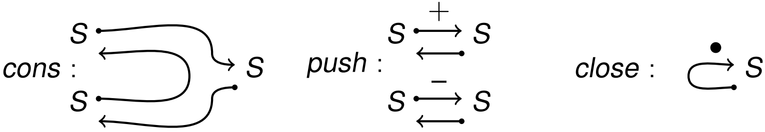
- Cowordisms for lists with deceptive slots:



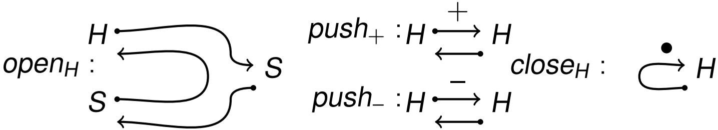
Subset sum problem

Generating solutions to SSP.

- We encode integers as words in the alphabet $\{+,-\}$, and integer lists as words in the alphabet $T = \{+,-,\bullet\}$, with $\bullet$ as a separation sign.
- Cowordisms for lists summing to zero:



- Cowordisms for lists with deceptive slots:



Replacing these with their names we get an honest LLG.

Conclusion

- **A detailed paper:** `https://arxiv.org/pdf/1911.03962.pdf`.
- **Current work:**
  The formalism of LLG extends to accommodate finer types, which allows conservative embedding of
  - Lambek grammars `https://arxiv.org/pdf/2005.10058.pdf`;
  - hopefully, some extensions (under study).
- **Thank you!**