Abstract Families of Abstract Categorial Languages

Makoto Kanazawa

National Institute of Informatics

Tokyo, Japan

This talk

- Closure properties of the languages generated by abstract categorial grammars (de Groote 2001).

This talk

- Closure properties of the languages generated by abstract categorial grammars (de Groote 2001).
- Each level $\boldsymbol{G}(m, n)$ of de Groote's hierarchy gives rise to a substitution-closed full abstract family of languages.

This talk

- Closure properties of the languages generated by abstract categorial grammars (de Groote 2001).
- Each level $\boldsymbol{G}(m, n)$ of de Groote's hierarchy gives rise to a substitution-closed full abstract family of languages.
- Most of the closure properties hold of the tree languages generated by ACGs, and more generally of the languages of λ-terms generated by ACGs.

This talk

- Closure properties of the languages generated by abstract categorial grammars (de Groote 2001).
- Each level $\boldsymbol{G}(m, n)$ of de Groote's hierarchy gives rise to a substitution-closed full abstract family of languages.
- Most of the closure properties hold of the tree languages generated by ACGs, and more generally of the languages of λ-terms generated by ACGs.
- Focuses on (a generalization of) closure under intersection with regular sets.

Outline

- Abstract categorial grammar: an informal idea
- Abstract categorial grammar: formal definitions and known results
- Closure under intersection with regular sets (generalized)

Abstract categorial grammar (de Groote 2001)

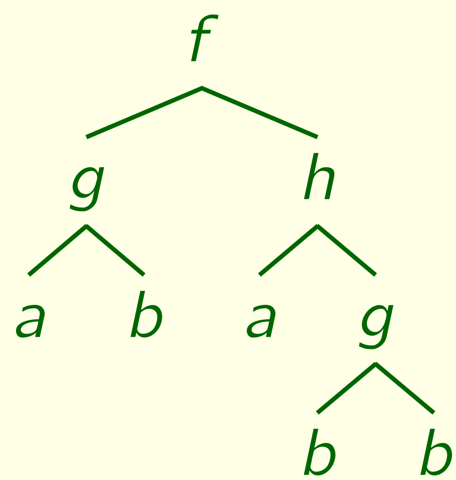
- A grammar formalism for languages of linear λ-terms.

Abstract categorial grammar (de Groote 2001)

- A grammar formalism for languages of linear λ-terms.
  - strings

$$ababb \qquad\qquad \lambda z.a(b(a(b(bz))))$$

  - trees



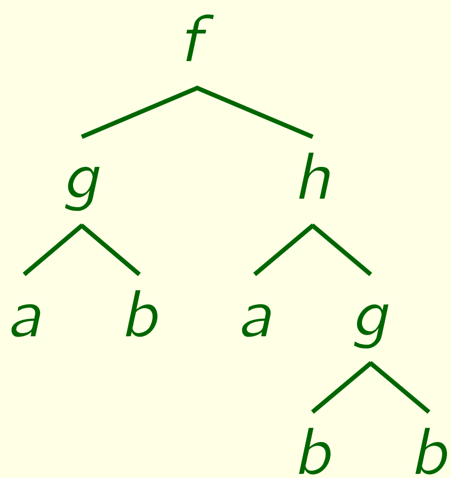
$$f(gab)(ha(gbb))$$

Abstract categorial grammar (de Groote 2001)

- A grammar formalism for languages of linear λ-terms.
  - strings

    $ababb \qquad \lambda z.a(b(a(b(bz))))$

  - trees



    $f(gab)(ha(gbb))$

  - and more
    - tuples of strings (trees)
    - logical formulae

Abstract categorial grammar (de Groote 2001)

- Generalizes
  - grammar formalisms with context-free derivation trees

Abstract categorial grammar (de Groote 2001)

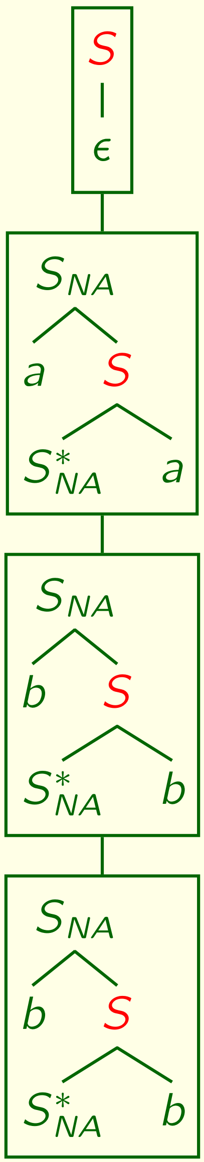
- Generalizes
  - grammar formalisms with context-free derivation trees
    - context-free grammar
    - multiple context-free grammar (linear context-free rewriting system)
    - tree-adjoining grammar

Abstract categorial grammar (de Groote 2001)

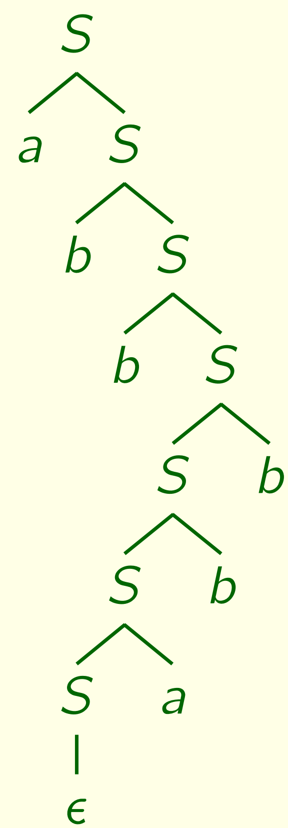
- Generalizes
  - grammar formalisms with context-free derivation trees
    - context-free grammar
    - multiple context-free grammar (linear context-free rewriting system)
    - tree-adjoining grammar
  - Montague semantics (modulo the linearity restriction)

Tree-adjoining grammar

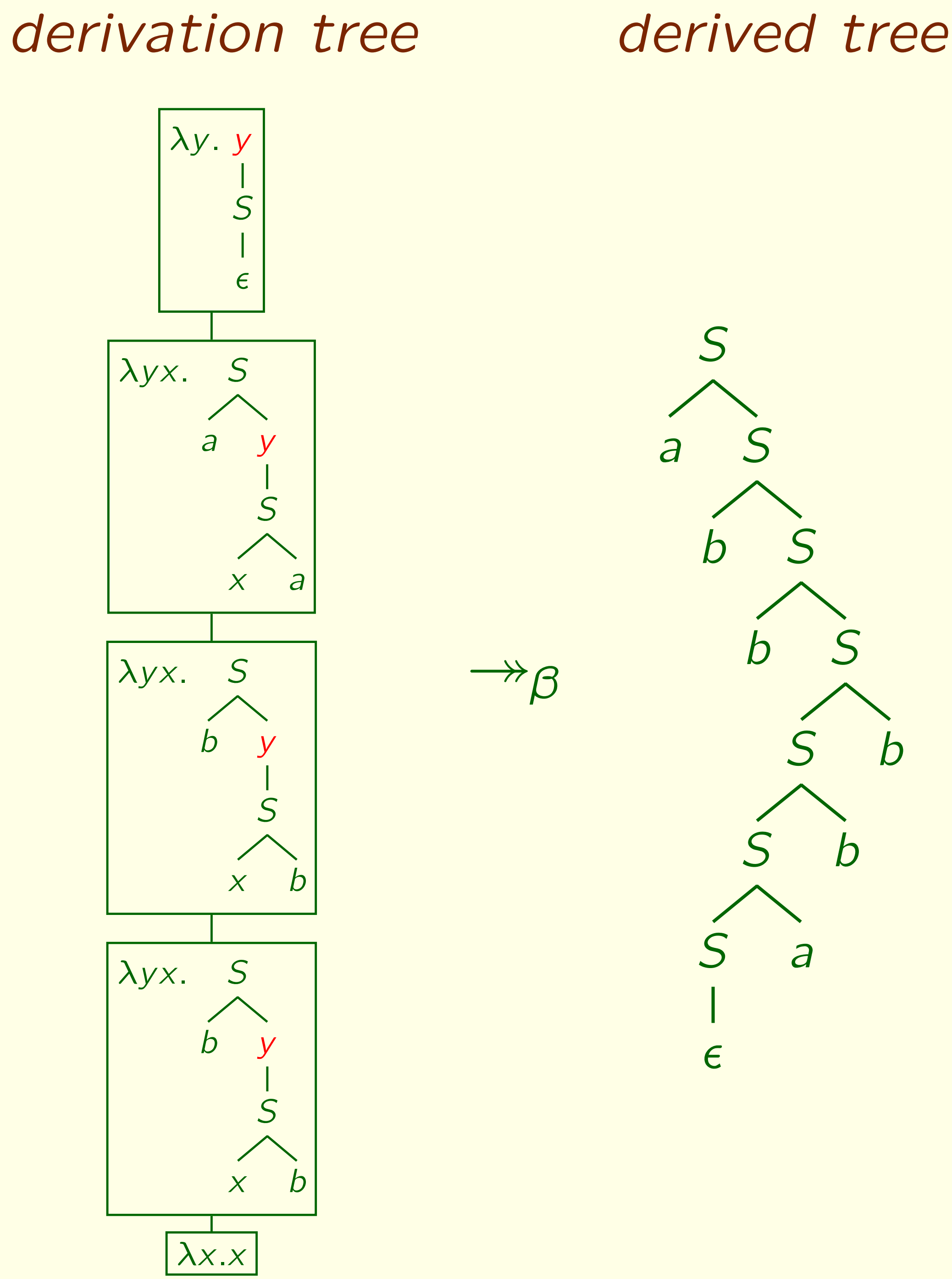
derivation tree *derived tree*



⟹



Tree-adjoining grammar as abstract categorial grammar

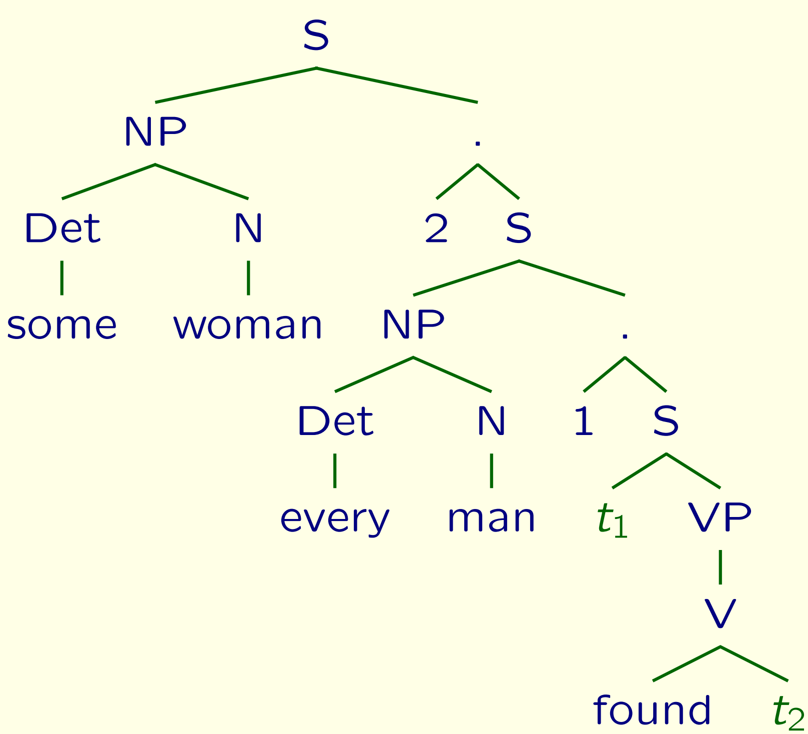


Montague semantics (Heim & Kratzer-style)

word meanings

$$[\![\text{some}]\!] = \lambda uv.\exists y.(uy \wedge vy)$$
$$[\![\text{every}]\!] = \lambda uv.\forall x.(ux \rightarrow vx)$$
$$[\![\text{woman}]\!] = \lambda y.\mathbf{woman}\ y$$
$$[\![\text{man}]\!] = \lambda x.\mathbf{man}\ x$$
$$[\![\text{found}]\!] = \lambda yx.\mathbf{find}\ y\ x$$

Logical Form



$$\Longrightarrow [\![\text{some}]\!][\![\text{woman}]\!](\lambda y.[\![\text{every}]\!][\![\text{man}]\!](\lambda x.[\![\text{found}]\!]yx))$$
$$\twoheadrightarrow_\beta \exists y.(\mathbf{woman}\ y \wedge \forall x.(\mathbf{man}\ x \rightarrow \mathbf{find}\ y\ x))$$

sentence meaning

Syntax and semantics with abstract categorial grammar

abstract derivation

$\textsc{some woman}\ (\lambda y.\textsc{every man}\ (\lambda x.\textsc{found}\ y\ x))$

$\xrightarrow{\beta}$ (sentence form) | $\xrightarrow{\beta}$ (sentence meaning)

$\textsc{some} = \lambda uv.v(\text{some} + u)$

$\textsc{every} = \lambda uv.v(\text{every} + u)$

$\textsc{woman} = \text{woman}$

$\textsc{man} = \text{man}$

$\textsc{found} = \lambda yx.x + \text{found} + y$

$\textsc{some} = \lambda uv.\exists y.(uy \wedge vy)$

$\textsc{every} = \lambda uv.\forall x.(ux \rightarrow vx)$

$\textsc{woman} = \lambda y.\mathbf{woman}\ y$

$\textsc{man} = \lambda x.\mathbf{man}\ x$

$\textsc{found} = \lambda yx.\mathbf{find}\ y\ x$

every + man + found + some + woman

sentence form

$\exists y.(\mathbf{woman}\ y \wedge \forall x.(\mathbf{man}\ x \rightarrow \mathbf{find}\ y\ x))$

sentence meaning

Syntax and semantics with abstract categorial grammar

abstract derivation

$$\text{SOME WOMAN}\ (\lambda y.\text{EVERY MAN}\ (\lambda x.\text{FOUND}\ y\ x))$$

$$\text{SOME} = \lambda uv.v(\text{some} + u)$$
$$\text{EVERY} = \lambda uv.v(\text{every} + u)$$
$$\text{WOMAN} = \text{woman}$$
$$\text{MAN} = \text{man}$$
$$\text{FOUND} = \lambda yx.x + \text{found} + y$$

$$\text{SOME} = \lambda uv.\exists y.(uy \land vy)$$
$$\text{EVERY} = \lambda uv.\forall x.(ux \to vx)$$
$$\text{WOMAN} = \lambda y.\mathbf{woman}\ y$$
$$\text{MAN} = \lambda x.\mathbf{man}\ x$$
$$\text{FOUND} = \lambda yx.\mathbf{find}\ y\ x$$

β β

$$\text{every} + \text{man} + \text{found} + \text{some} + \text{woman}$$

sentence form

$$\exists y.(\mathbf{woman}\ y \land \forall x.(\mathbf{man}\ x \to \mathbf{find}\ y\ x))$$

sentence meaning

$$M + N = \lambda z.M(Nz)$$

Higher-order signature

$$\Sigma = \langle A, C, \tau \rangle$$

- A is a set of atomic types
- C is a set of constants
- $\tau : C \to \mathscr{T}(A)$ (type assignment to constants)

$\mathscr{T}(A)$ is the set of types built up from A with $\to$:

$$\alpha, \beta \in \mathscr{T}(A) \Longrightarrow \alpha \to \beta \in \mathscr{T}(A).$$

Write $\alpha \to \beta \to \gamma$ for $\alpha \to (\beta \to \gamma)$.

Higher-order signature

$$\Sigma = \langle A, C, \tau \rangle$$

- A is a set of atomic types
- C is a set of constants
- $\tau \colon C \to \mathscr{T}(A)$ (type assignment to constants)

$\mathscr{T}(A)$ is the set of types built up from A with $\to$:

$$\alpha, \beta \in \mathscr{T}(A) \Longrightarrow \alpha \to \beta \in \mathscr{T}(A).$$

Write $\alpha \to \beta \to \gamma$ for $\alpha \to (\beta \to \gamma)$.

Define the order of a type:

$$\mathrm{ord}(p) = 1 \quad \text{if } p \text{ is atomic,}$$
$$\mathrm{ord}(\alpha \to \beta) = \max(\mathrm{ord}(\alpha) + 1, \mathrm{ord}(\beta)).$$

The order of Σ is $\mathrm{ord}(\Sigma) = \max\{\, \mathrm{ord}(\tau(c)) \mid c \in C \,\}$.

λ-terms over Σ

$\Lambda(\Sigma)$ consists of

- $x \in X$ (variable),
- $c \in C$,
- MN for $M, N \in \Lambda(\Sigma)$,
- $\lambda x.M$ for $x \in X$, $M \in \Lambda(\Sigma)$.

λ-terms over Σ

$\Lambda(\Sigma)$ consists of

- $x \in X$ (variable),
- $c \in C$,
- MN for $M, N \in \Lambda(\Sigma)$,
- $\lambda x.M$ for $x \in X$, $M \in \Lambda(\Sigma)$.

Write

$$\begin{aligned} MNP &\quad \text{for} \quad (MN)P, \\ \lambda x.MN &\quad \text{for} \quad \lambda x.(MN), \\ \lambda x_1 \ldots x_n.M &\quad \text{for} \quad \lambda x_1.(\lambda x_2. \ldots (\lambda x_n.M) \ldots). \end{aligned}$$

λ-terms over Σ

$$\begin{aligned} FV(x) &= \{x\}, \\ FV(c) &= \varnothing, \\ FV(MN) &= FV(M) \cup FV(N), \\ FV(\lambda x.M) &= FV(M) - \{x\}. \end{aligned}$$

λ-terms over Σ

$$
\begin{aligned}
FV(x) &= \{x\}, \\
FV(c) &= \varnothing, \\
FV(MN) &= FV(M) \cup FV(N), \\
FV(\lambda x.M) &= FV(M) - \{x\}.
\end{aligned}
$$

$$
\begin{aligned}
\overrightarrow{Con}(x) &= \epsilon, \\
\overrightarrow{Con}(c) &= c, \\
\overrightarrow{Con}(MN) &= \overrightarrow{Con}(M)\overrightarrow{Con}(N), \\
\overrightarrow{Con}(\lambda x.M) &= \overrightarrow{Con}(M).
\end{aligned}
$$

λ-terms over Σ

$$
\begin{aligned}
FV(x) &= \{x\},\\
FV(c) &= \varnothing,\\
FV(MN) &= FV(M) \cup FV(N),\\
FV(\lambda x.M) &= FV(M) - \{x\}.
\end{aligned}
$$

$$
\begin{aligned}
\overrightarrow{Con}(x) &= \epsilon,\\
\overrightarrow{Con}(c) &= c,\\
\overrightarrow{Con}(MN) &= \overrightarrow{Con}(M)\overrightarrow{Con}(N),\\
\overrightarrow{Con}(\lambda x.M) &= \overrightarrow{Con}(M).
\end{aligned}
$$

M is

- closed if $FV(M) = \varnothing$
- pure if $\overrightarrow{Con}(M) = \epsilon$.

β-reduction

$$\ldots(\lambda x.M)N\cdots \to_\beta \ldots M[x := N]\ldots$$

This β-reduction step is

- non-erasing if $x \in FV(M)$,
- non-duplicating if x occurs free in M at most once.

Write $|M|_\beta$ for the β-normal form M.

Type assignment system $\lambda{\to}_\Sigma$

$\Gamma = x_1 : \alpha_1, \ldots, x_n : \alpha_n$: type environment

$\Gamma \vdash_\Sigma M : \alpha$: typing judgment

$$\vdash_\Sigma c : \tau(c) \qquad x : \alpha \vdash_\Sigma x : \alpha$$

$$\frac{\Gamma \vdash_\Sigma M : \beta}{\Gamma - \{x : \alpha\} \vdash_\Sigma \lambda x.M : \alpha \to \beta} \to I \qquad \frac{\Gamma \vdash_\Sigma M : \alpha \to \beta \quad \Delta \vdash_\Sigma N : \alpha}{\Gamma \cup \Delta \vdash_\Sigma MN : \beta} \to E$$

Write Λ and $\vdash$ for $\Lambda(\Sigma)$ and $\vdash_\Sigma$ when $\Sigma = \langle A, \varnothing, \varnothing \rangle$.

Linear λ-terms

The set $\Lambda_{\mathrm{lin}}(\Sigma)$ of linear λ-terms consists of λ-terms $M \in \Lambda(\Sigma)$ such that

(i) for every subterm $\lambda x.N$ of M, $x \in FV(N)$,

(ii) for every subterm NP of M, $FV(N) \cap FV(P) = \varnothing$.

Linear λ-terms

The set $\Lambda_{\text{lin}}(\Sigma)$ of linear λ-terms consists of λ-terms $M \in \Lambda(\Sigma)$ such that

(i) for every subterm $\lambda x.N$ of M, $x \in FV(N)$,

(ii) for every subterm NP of M, $FV(N) \cap FV(P) = \varnothing$.

M is a λI-term if it satisfies (i).

Linear λ-terms

The set $\Lambda_{\text{lin}}(\Sigma)$ of linear λ-terms consists of λ-terms $M \in \Lambda(\Sigma)$ such that

(i) for every subterm $\lambda x.N$ of M, $x \in FV(N)$,

(ii) for every subterm NP of M, $FV(N) \cap FV(P) = \varnothing$.

M is a λI-term if it satisfies (i).

Strings and concatenation of strings are represented by linear λ-terms.

$$/a_1 \dots a_n/ = \lambda z.a_1(\dots(a_n z)\dots)$$

$$\mathbf{+} = \lambda xyz.x(yz)$$

Linear λ-terms

The set $\Lambda_{\text{lin}}(\Sigma)$ of linear λ-terms consists of λ-terms $M \in \Lambda(\Sigma)$ such that

(i) for every subterm $\lambda x.N$ of M, $x \in FV(N)$,

(ii) for every subterm NP of M, $FV(N) \cap FV(P) = \varnothing$.

M is a λI-term if it satisfies (i).

Strings and concatenation of strings are represented by linear λ-terms.

$$/a_1 \dots a_n/ = \lambda z.a_1(\dots(a_n z)\dots)$$

$$\mathbf{+} = \lambda xyz.x(yz)$$

String signature $\Sigma_V = \langle \{o\}, V, \tau \rangle$:

$$\tau(a) = o \to o = str \quad \text{for every } a \in V,$$

$$\vdash_{\Sigma_V} /w/ : str \quad \text{for every } w \in V^*.$$

Abstract categorial grammar

$$\mathscr{G} = \langle \Sigma, \Sigma', \mathscr{L}, s \rangle$$

- $\Sigma = \langle A, C, \tau \rangle$: higher-order signature (abstract vocabulary)
- $\Sigma' = \langle A', C', \tau' \rangle$: higher-order signature (object vocabulary)
- $\mathscr{L} = \langle \sigma, \theta \rangle$: lexicon from Σ to Σ':
  - $\sigma : A \to \mathscr{T}(A')$,
  - $\theta : C \to \Lambda_{\text{lin}}(\Sigma')$,
  - $\vdash_{\Sigma'} \theta(c) : \sigma(\tau(c))$ for every $c \in C$.
- s: atomic type of Σ (distinguished type).

Abstract categorial grammar

$$\mathscr{G} = \langle \Sigma, \Sigma', \mathscr{L}, s \rangle$$

- $\Sigma = \langle A, C, \tau \rangle$: higher-order signature (abstract vocabulary)
- $\Sigma' = \langle A', C', \tau' \rangle$: higher-order signature (object vocabulary)
- $\mathscr{L} = \langle \sigma, \theta \rangle$: lexicon from Σ to Σ':
  - $\sigma : A \to \mathscr{T}(A')$,
  - $\theta : C \to \Lambda_{\mathrm{lin}}(\Sigma')$,
  - $\vdash_{\Sigma'} \theta(c) : \sigma(\tau(c))$ for every $c \in C$.
- s: atomic type of Σ (distinguished type).

θ is naturally extended to a mapping from $\Lambda_{\mathrm{lin}}(\Sigma)$ to $\Lambda_{\mathrm{lin}}(\Sigma')$.

Write $\mathscr{L}(\alpha)$ and $\mathscr{L}(M)$ for $\sigma(\alpha)$ and $\theta(M)$, respectively.

Abstract categorial grammar

$$\mathscr{G} = \langle \Sigma, \Sigma', \mathscr{L}, s \rangle$$

- $\Sigma = \langle A, C, \tau \rangle$: higher-order signature (abstract vocabulary)
- $\Sigma' = \langle A', C', \tau' \rangle$: higher-order signature (object vocabulary)
- $\mathscr{L} = \langle \sigma, \theta \rangle$: lexicon from Σ to Σ':
  - $\sigma : A \to \mathscr{T}(A')$,
  - $\theta : C \to \Lambda_{\mathrm{lin}}(\Sigma')$,
  - $\vdash_{\Sigma'} \theta(c) : \sigma(\tau(c))$ for every $c \in C$.
- s: atomic type of Σ (distinguished type).

θ is naturally extended to a mapping from $\Lambda_{\mathrm{lin}}(\Sigma)$ to $\Lambda_{\mathrm{lin}}(\Sigma')$.

Write $\mathscr{L}(\alpha)$ and $\mathscr{L}(M)$ for $\sigma(\alpha)$ and $\theta(M)$, respectively.

The order of $\mathscr{L}$ is $\mathrm{ord}(\mathscr{L}) = \max\{\, \mathrm{ord}(\mathscr{L}(p)) \mid p \in A \,\}$.

Abstract categorial grammar

$$\mathscr{G} = \langle \Sigma, \Sigma', \mathscr{L}, s \rangle$$

- $\Sigma = \langle A, C, \tau \rangle$: higher-order signature (abstract vocabulary)
- $\Sigma' = \langle A', C', \tau' \rangle$: higher-order signature (object vocabulary)
- $\mathscr{L} = \langle \sigma, \theta \rangle$: lexicon from Σ to Σ':
  - $\sigma : A \to \mathscr{T}(A')$,
  - $\theta : C \to \Lambda_{\mathrm{lin}}(\Sigma')$,
  - $\vdash_{\Sigma'} \theta(c) : \sigma(\tau(c))$ for every $c \in C$.
- s: atomic type of Σ (distinguished type).

θ is naturally extended to a mapping from $\Lambda_{\mathrm{lin}}(\Sigma)$ to $\Lambda_{\mathrm{lin}}(\Sigma')$.

Write $\mathscr{L}(\alpha)$ and $\mathscr{L}(M)$ for $\sigma(\alpha)$ and $\theta(M)$, respectively.

The order of $\mathscr{L}$ is $\mathrm{ord}(\mathscr{L}) = \max\{\, \mathrm{ord}(\mathscr{L}(p)) \mid p \in A \,\}$.

$\mathscr{G} \in \boldsymbol{G}(m, n)$ if $\mathrm{ord}(\Sigma) \leq m$ and $\mathrm{ord}(\mathscr{L}) \leq n$.

Abstract categorial grammar

$$\mathscr{G} = \langle \Sigma, \Sigma', \mathscr{L}, s \rangle$$

- $\Sigma = \langle A, C, \tau \rangle$: higher-order signature (abstract vocabulary)
- $\Sigma' = \langle A', C', \tau' \rangle$: higher-order signature (object vocabulary)
- $\mathscr{L} = \langle \sigma, \theta \rangle$: lexicon from Σ to Σ':
  - $\sigma : A \to \mathscr{T}(A')$,
  - $\theta : C \to \Lambda_{\text{lin}}(\Sigma')$,
  - $\vdash_{\Sigma'} \theta(c) : \sigma(\tau(c))$ for every $c \in C$.
- s: atomic type of Σ (distinguished type).

θ is naturally extended to a mapping from $\Lambda_{\text{lin}}(\Sigma)$ to $\Lambda_{\text{lin}}(\Sigma')$.

Write $\mathscr{L}(\alpha)$ and $\mathscr{L}(M)$ for $\sigma(\alpha)$ and $\theta(M)$, respectively.

The order of $\mathscr{L}$ is $\text{ord}(\mathscr{L}) = \max\{\, \text{ord}(\mathscr{L}(p)) \mid p \in A \,\}$.

$\mathscr{G} \in \boldsymbol{G}(m, n)$ if $\text{ord}(\Sigma) \leq m$ and $\text{ord}(\mathscr{L}) \leq n$.

$\mathscr{G}$ is m-th order if $\mathscr{G} \in \boldsymbol{G}(m, n)$ for some n.

Languages of ACGs

The abstract language of $\mathscr{G}$ is

$$\mathcal{A}(\mathscr{G}) = \{\, M \in \Lambda_{\mathsf{lin}}(\Sigma) \mid M \text{ is } \beta\text{-normal and } \vdash_{\Sigma} M : s \,\}.$$

the set of abstract derivations

The object language of $\mathscr{G}$ is

$$\mathcal{O}(\mathscr{G}) = \{\, |\mathscr{L}(M)|_{\beta} \mid M \in \mathcal{A}(\mathscr{G}) \,\}.$$

the set of concrete forms

Languages of ACGs

The abstract language of $\mathscr{G}$ is

$$\mathcal{A}(\mathscr{G}) = \{\, M \in \Lambda_{\mathsf{lin}}(\Sigma) \mid M \text{ is } \beta\text{-normal and } \vdash_{\Sigma} M : s \,\}.$$

the set of abstract derivations

The object language of $\mathscr{G}$ is

$$\mathcal{O}(\mathscr{G}) = \{\, |\mathscr{L}(M)|_{\beta} \mid M \in \mathcal{A}(\mathscr{G}) \,\}.$$

the set of concrete forms

We say that $\mathscr{G}$ generates its object language.

Example

$c \in C$	$\tau(c)$	$\mathscr{L}(c)$	$\mathscr{L}(\tau(c))$
A	$(p_1 \to s) \to s$	$\lambda u.\ /a/ + u/\epsilon/$	$(str \to str) \to str$
B	$(p_2 \to s) \to s$	$\lambda u.\ /b/ + u/\epsilon/$	$(str \to str) \to str$
C	$(p_3 \to s) \to s$	$\lambda u.\ /c/ + u/\epsilon/$	$(str \to str) \to str$
D	$q \to s$	$\lambda v.v$	$str \to str$
E	$p_1 \to p_2 \to p_3 \to q \to q$	$\lambda x_1 x_2 x_3 v.\ x_1 + x_2 + x_3 + v$	$str \to str \to str \to str \to str$
F	q	$/\epsilon/$	str

Example

$c \in C$	$\tau(c)$	$\mathscr{L}(c)$	$\mathscr{L}(\tau(c))$
A	$(p_1 \to s) \to s$	$\lambda u.\ /a/ + u/\epsilon/$	$(str \to str) \to str$
B	$(p_2 \to s) \to s$	$\lambda u.\ /b/ + u/\epsilon/$	$(str \to str) \to str$
C	$(p_3 \to s) \to s$	$\lambda u.\ /c/ + u/\epsilon/$	$(str \to str) \to str$
D	$q \to s$	$\lambda v.v$	$str \to str$
E	$p_1 \to p_2 \to p_3 \to q \to q$	$\lambda x_1 x_2 x_3 v.\ x_1 + x_2 + x_3 + v$	$str \to str \to str \to str \to str$
F	q	$/\epsilon/$	str

$$\mathscr{G} \in \boldsymbol{G}(3, 2).$$

Example

$c \in C$	$\tau(c)$	$\mathscr{L}(c)$	$\mathscr{L}(\tau(c))$
A	$(p_1 \to s) \to s$	$\lambda u.\ /a/ + u/\epsilon/$	$(str \to str) \to str$
B	$(p_2 \to s) \to s$	$\lambda u.\ /b/ + u/\epsilon/$	$(str \to str) \to str$
C	$(p_3 \to s) \to s$	$\lambda u.\ /c/ + u/\epsilon/$	$(str \to str) \to str$
D	$q \to s$	$\lambda v.v$	$str \to str$
E	$p_1 \to p_2 \to p_3 \to q \to q$	$\lambda x_1 x_2 x_3 v.\ x_1 + x_2 + x_3 + v$	$str \to str \to str \to str \to str$
F	q	$/\epsilon/$	str

$$\mathscr{G} \in \boldsymbol{G}(3, 2).$$

$$P = \mathrm{A}(\lambda x_1.\mathrm{B}(\lambda y_1.\mathrm{B}(\lambda y_2.\mathrm{A}(\lambda x_2.\mathrm{C}(\lambda z_1.\mathrm{C}(\lambda z_2.\mathrm{D}(\mathrm{E} x_1 y_1 z_1(\mathrm{E} x_2 y_2 z_2 \mathrm{F}))))))))) \in \mathcal{A}(\mathscr{G}),$$

$$\mathscr{L}(P) \twoheadrightarrow_\beta /abbacc/ \in \mathcal{O}(\mathscr{G}).$$

Example

$c \in C$	$\tau(c)$	$\mathscr{L}(c)$	$\mathscr{L}(\tau(c))$
A	$(p_1 \to s) \to s$	$\lambda u.\ /a/ + u/\epsilon/$	$(str \to str) \to str$
B	$(p_2 \to s) \to s$	$\lambda u.\ /b/ + u/\epsilon/$	$(str \to str) \to str$
C	$(p_3 \to s) \to s$	$\lambda u.\ /c/ + u/\epsilon/$	$(str \to str) \to str$
D	$q \to s$	$\lambda v.v$	$str \to str$
E	$p_1 \to p_2 \to p_3 \to q \to q$	$\lambda x_1 x_2 x_3 v.\ x_1 + x_2 + x_3 + v$	$str \to str \to str \to str \to str$
F	q	$/\epsilon/$	str

$$\mathscr{G} \in \boldsymbol{G}(3,2).$$

$$P = \text{A}(\lambda x_1.\text{B}(\lambda y_1.\text{B}(\lambda y_2.\text{A}(\lambda x_2.\text{C}(\lambda z_1.\text{C}(\lambda z_2.\text{D}(\text{E}x_1 y_1 z_1(\text{E}x_2 y_2 z_2\text{F}))))))))) \in \mathcal{A}(\mathscr{G}),$$

$$\mathscr{L}(P) \twoheadrightarrow_\beta /abbacc/ \in \mathcal{O}(\mathscr{G}).$$

$$\mathcal{O}(\mathscr{G}) = \{\, /w/ \mid w \in \text{MIX}\,\}, \quad \text{where}$$

$$\text{MIX} = \{\, w \in \{a,b,c\}^* \mid \#_a(w) = \#_b(w) = \#_c(w)\,\}.$$

Complexity

NON-EMPTINESS

- Instance: An ACG $\mathscr{G}$.
- Question: Is $\mathcal{O}(\mathscr{G})$ (or, equivalently, $\mathcal{A}(\mathscr{G})$) non-empty?

Complexity

NON-EMPTINESS

- Instance: An ACG $\mathscr{G}$.
- Question: Is $\mathcal{O}(\mathscr{G})$ (or, equivalently, $\mathcal{A}(\mathscr{G})$) non-empty?

UNIVERSAL RECOGNITION

- Instance: An ACG $\mathscr{G} = \langle \Sigma, \Sigma', \mathscr{L}, s \rangle$ and $M \in \Lambda(\Sigma')$.
- Question: $M \in \mathcal{O}(\mathscr{L})$?

Complexity

NON-EMPTINESS

- Instance: An ACG $\mathscr{G}$.
- Question: Is $\mathcal{O}(\mathscr{G})$ (or, equivalently, $\mathcal{A}(\mathscr{G})$) non-empty?

UNIVERSAL RECOGNITION

- Instance: An ACG $\mathscr{G} = \langle \Sigma, \Sigma', \mathscr{L}, s \rangle$ and $M \in \Lambda(\Sigma')$.
- Question: $M \in \mathcal{O}(\mathscr{L})$?

NON-EMPTINESS

- is decidalbe if and only if **MELL** is decidadble;
- is at least EXPSPACE-hard;
- reduces to UNIVERSAL RECOGNITION.

Complexity

NON-EMPTINESS

- Instance: An ACG $\mathscr{G}$.
- Question: Is $\mathcal{O}(\mathscr{G})$ (or, equivalently, $\mathcal{A}(\mathscr{G})$) non-empty?

UNIVERSAL RECOGNITION

- Instance: An ACG $\mathscr{G} = \langle \Sigma, \Sigma', \mathscr{L}, s \rangle$ and $M \in \Lambda(\Sigma')$.
- Question: $M \in \mathcal{O}(\mathscr{L})$?

NON-EMPTINESS

- is decidalbe if and only if **MELL** is decidadble;
- is at least EXPSPACE-hard;
- reduces to UNIVERSAL RECOGNITION.

Both problems are NP-complete when restricted to lexicalized ACGs (follows from the NP-completeness of **MLL**($\multimap$)).

Generative capacity

$$\mathscr{G} \in \boldsymbol{G}(2, n)$$

n	string languages	tree languages
1	/	REGT
2	CF	$\mathrm{CFT}_{\mathrm{sp}}$
3	$\mathrm{yCFT}_{\mathrm{sp}}$	$\supsetneq$ MREGT
≥ 4	MCF = STR(HR)	TR(HR)

Generative capacity

$$\mathscr{G} \in \boldsymbol{G}(2, n)$$

n	string languages	tree languages
1		REGT
2	CF	$\mathrm{CFT}_{\mathrm{sp}}$
3	$\mathrm{yCFT}_{\mathrm{sp}}$	$\supsetneq$ MREGT
≥ 4	MCF = STR(HR)	TR(HR)

These languages are semilinear and belong to LOGCFL.

Generative capacity

$$\mathscr{G} \in \boldsymbol{G}(2, n)$$

n	string languages	tree languages
1		REGT
2	CF	$\mathrm{CFT}_{\mathrm{sp}}$
3	$\mathrm{yCFT}_{\mathrm{sp}}$	$\supsetneq$ MREGT
≥ 4	MCF = STR(HR)	TR(HR)

These languages are semilinear and belong to LOGCFL.

Not much is known for higher-order cases:

- $\boldsymbol{G}(3, 2)$: non-semilinear string languages.
- $\boldsymbol{G}(3, 1)$: NP-complete tree languages.
- No example of an r.e. language has been found that cannot be generated by an ACG.

ACGs and AFLs

The string languages generated by ACGs in $\boldsymbol{G}(m,n)$ $(m,n \geq 2)$ form a substitution-closed full AFL.

ACGs and AFLs

The string languages generated by ACGs in $\boldsymbol{G}(m, n)$ ($m, n \geq 2$) form a substitution-closed full AFL.

A family of languages is a full abstract family of languages if it is closed under

- union ($\cup$), concatenation ($\cdot$), Kleene star (*);
- homomorphism (h);
- inverse homomorphism (h^{-1});
- intersection with regular sets ($\cap R$).

ACGs and AFLs

The string languages generated by ACGs in $\boldsymbol{G}(m, n)$ $(m, n \geq 2)$ form a substitution-closed full AFL.

A family of languages is a full abstract family of languages if it is closed under

- union ($\cup$), concatenation ($\cdot$), Kleene star (*);
- homomorphism (h);
- inverse homomorphism (h^{-1});
- intersection with regular sets ($\cap R$).

Why is this interesting?

ACGs and AFLs

The string languages generated by ACGs in $\boldsymbol{G}(m, n)$ $(m, n \geq 2)$ form a substitution-closed full AFL.

A family of languages is a full abstract family of languages if it is closed under

- union ($\cup$), concatenation ($\cdot$), Kleene star (*);
- homomorphism (h);
- inverse homomorphism (h^{-1});
- intersection with regular sets ($\cap R$).

Why is this interesting?

- Not entirely obvious ($\cap R$).
- Depends on some techinical results about $\lambda{\rightarrow}_{\Sigma}$.
- Hopefully useful.
- May lead to an automaton model for ACGs.

Important facts about $\lambda{\rightarrow}_{\Sigma}$

Subject Reduction Theorem.

If $\Gamma \vdash_{\Sigma} M : \alpha$ and $M \twoheadrightarrow_{\beta} M'$, then $\Gamma \vdash_{\Sigma} M' : \alpha$.

Important facts about $\lambda{\rightarrow}_{\Sigma}$

Subject Reduction Theorem.

If $\Gamma \vdash_{\Sigma} M : \alpha$ and $M \twoheadrightarrow_{\beta} M'$, then $\Gamma \vdash_{\Sigma} M' : \alpha$.

Subject Expansion Theorem.

If $\Gamma \vdash_{\Sigma} M' : \alpha$ and $M \twoheadrightarrow_{\beta} M'$ by non-erasing non-duplicating β-reduction, then $\Gamma \vdash_{\Sigma} M : \alpha$.

Important facts about $\lambda{\to}_\Sigma$

Subject Reduction Theorem.

If $\Gamma \vdash_\Sigma M : \alpha$ and $M \twoheadrightarrow_\beta M'$, then $\Gamma \vdash_\Sigma M' : \alpha$.

Subject Expansion Theorem.

If $\Gamma \vdash_\Sigma M' : \alpha$ and $M \twoheadrightarrow_\beta M'$ by non-erasing non-duplicating β-reduction, then $\Gamma \vdash_\Sigma M : \alpha$.

(A special case: M linear.)

Important facts about $\lambda{\rightarrow}_{\Sigma}$

Subject Reduction Theorem.

If $\Gamma \vdash_{\Sigma} M : \alpha$ and $M \twoheadrightarrow_{\beta} M'$, then $\Gamma \vdash_{\Sigma} M' : \alpha$.

Subject Expansion Theorem.

If $\Gamma \vdash_{\Sigma} M' : \alpha$ and $M \twoheadrightarrow_{\beta} M'$ by non-erasing non-duplicating β-reduction, then $\Gamma \vdash_{\Sigma} M : \alpha$.
(A special case: M linear.)

Uniqueness Theorem.

If M is a λI-term and $\Gamma \vdash_{\Sigma} M : \alpha$, then there is a unique $\lambda{\rightarrow}_{\Sigma}$-deduction of this judgment.

Important facts about $\lambda{\to}_\Sigma$

Subject Reduction Theorem.
If $\Gamma \vdash_\Sigma M : \alpha$ and $M \twoheadrightarrow_\beta M'$, then $\Gamma \vdash_\Sigma M' : \alpha$.

Subject Expansion Theorem.
If $\Gamma \vdash_\Sigma M' : \alpha$ and $M \twoheadrightarrow_\beta M'$ by non-erasing non-duplicating β-reduction, then $\Gamma \vdash_\Sigma M : \alpha$.
(A special case: M linear.)

Uniqueness Theorem.
If M is a λI-term and $\Gamma \vdash_\Sigma M : \alpha$, then there is a unique $\lambda{\to}_\Sigma$-deduction of this judgment.

Principal Pair Theorem.
If $\Gamma \vdash M : \alpha$ then there is a most general such $\langle \Gamma, \alpha \rangle$ (called a principal pair for M).

Properties of lexicons

β-reduction commutes with lexicons:

$$M \twoheadrightarrow_\beta M' \quad \text{implies} \quad \mathscr{L}(M) \twoheadrightarrow_\beta \mathscr{L}(M').$$

Properties of lexicons

β-reduction commutes with lexicons:

$$M \twoheadrightarrow_\beta M' \quad \text{implies} \quad \mathscr{L}(M) \twoheadrightarrow_\beta \mathscr{L}(M').$$

Typing judgments are preserved under lexicons:

$$\Gamma \vdash_\Sigma M : \alpha \quad \text{implies} \quad \mathscr{L}(\Gamma) \vdash_{\Sigma'} \mathscr{L}(M) : \mathscr{L}(\alpha).$$

Properties of lexicons

β-reduction commutes with lexicons:

$$M \twoheadrightarrow_\beta M' \quad \text{implies} \quad \mathscr{L}(M) \twoheadrightarrow_\beta \mathscr{L}(M').$$

Typing judgments are preserved under lexicons:

$$\Gamma \vdash_\Sigma M : \alpha \quad \text{implies} \quad \mathscr{L}(\Gamma) \vdash_{\Sigma'} \mathscr{L}(M) : \mathscr{L}(\alpha).$$

If $\mathscr{L}_1 = \langle \sigma_1, \theta_1 \rangle$ is a lexicon from Σ_0 to Σ_1 and $\mathscr{L}_2 = \langle \sigma_2, \theta_2 \rangle$ is a lexicon from Σ_1 to Σ_2, then

$$\mathscr{L}_2 \circ \mathscr{L}_1 = \langle \sigma_2 \circ \sigma_1, \theta_2 \circ \theta_1 \rangle$$

is a lexicon from Σ_0 to Σ_2.

Relabeling

$$\mathscr{L} : \Sigma \to \Sigma'$$

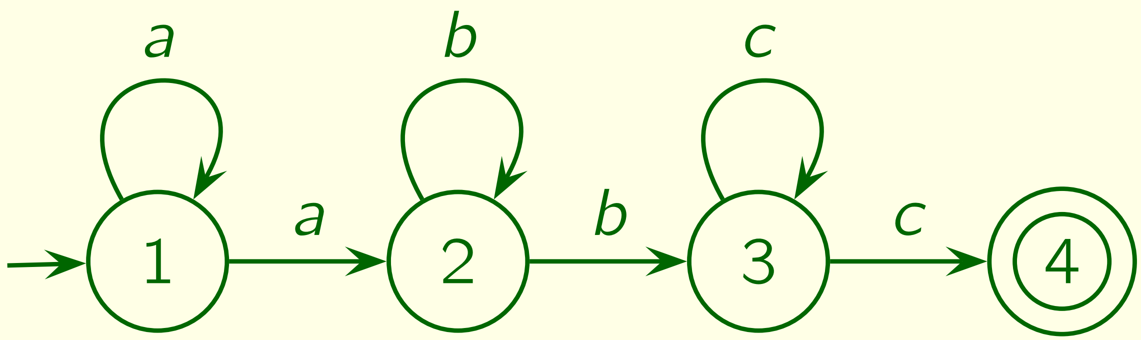
- $\mathscr{L}(p) \in A'$ for all $p \in A$
- $\mathscr{L}(c) \in C'$ for all $c \in C$

Relabeling

$$\mathscr{L} : \Sigma \to \Sigma'$$

- $\mathscr{L}(p) \in A'$ for all $p \in A$
- $\mathscr{L}(c) \in C'$ for all $c \in C$

A nondeterministic finite automaton $\boldsymbol{M} = \langle Q, V, \delta, q_I, \{q_F\}\rangle$:



$$A = Q, \qquad \mathscr{L}(p) = o \quad \text{for all } p \in a,$$

$$C = \{\, d^{r \to q} \mid r \in \delta(q, d)\,\}, \quad \mathscr{L}(d^{r \to q}) = d.$$

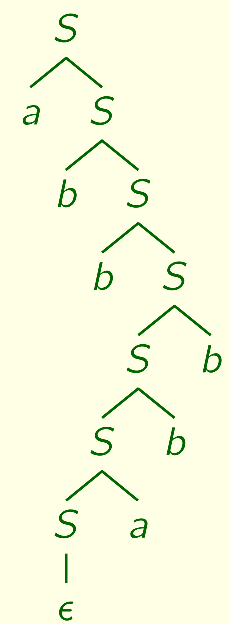
$$\tau(d^{r \to q}) = r \to q.$$

$$w \in L(\boldsymbol{M}) \iff /w/ \in \{\, \mathscr{L}(N) \mid \vdash_\Sigma N : q_F \to q_I \,\}.$$

Relabeling

A nondeterministic bottom-up finite tree automaton
$M = \langle Q, F, \{q_F\}, \delta \rangle$:

$a \to 1$	$S52 \to 5$
$b \to 2$	$S52 \to 6$
$\epsilon \to 3$	$S26 \to 6$
$S3 \to 4$	$S26 \to 7$
$S41 \to 4$	$S17 \to 7$
$S41 \to 5$	



$$A = Q,$$
$$C = \{ d^{q_1 \to \cdots \to q_n \to r} \mid d q_1 \ldots q_n \to r \in \delta \},$$
$$\tau(d^{q_1 \to \cdots \to q_n \to r}) = q_1 \to \cdots \to q_n \to r,$$
$$\mathscr{L}(p) = o \quad \text{for all } p \in A,$$
$$\mathscr{L}(d^{q_1 \to \cdots \to q_n \to r}) = d.$$

$$T \in L(M) \Longleftrightarrow T \in \{ \mathscr{L}(N) \mid \vdash_\Sigma N : q_F \}$$

Intersection with the image of a relabeling

ACG $\mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle$ relabeling $\mathscr{L}_1 \colon \Sigma_1' \to \Sigma_1$

type $\boldsymbol{\gamma} \in \mathscr{T}(A')$

Construct

$$\mathscr{G}_\cap = \langle \Sigma_0', \Sigma_1, \mathscr{L}_1 \circ \mathscr{L}', s^\gamma \rangle$$

such that

$$\mathcal{O}(\mathscr{G}_\cap) = \mathcal{O}(\mathscr{G}) \cap \{ \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \}.$$

Intersection with the image of a relabeling

$$\text{ACG } \mathscr{G} = \langle \Sigma_0, \Sigma_1, \mathscr{L}, s \rangle \quad \text{relabeling } \mathscr{L}_1 \colon \Sigma_1' \to \Sigma_1$$

$$\text{type } \boldsymbol{\gamma} \in \mathscr{T}(A')$$

Construct

$$\mathscr{G}_\cap = \langle \Sigma_0', \Sigma_1, \mathscr{L}_1 \circ \mathscr{L}', s^\gamma \rangle$$

such that

$$\mathcal{O}(\mathscr{G}_\cap) = \mathcal{O}(\mathscr{G}) \cap \{ \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \}.$$

- The construction generalizes standard constructions for well-known grammar formalisms,
- but the proof of correctness is a lot more involved due to its generality.

Construction

$$\mathscr{G}_\cap = \langle \Sigma_0', \Sigma_1, \mathscr{L}_1 \circ \mathscr{L}', s^\gamma \rangle$$

$\Sigma_0' = \langle A_0', C_0', \tau_0' \rangle$:

$$A_0' = \{\, p^\beta \mid p \in A_0, \beta \in \mathscr{T}(A_1'), \mathscr{L}_1(\beta) = \mathscr{L}(p) \,\},$$

$$C_0' = \{\, d_{\langle c, N, \beta \rangle} \mid c \in C_0, N \in \Lambda_{\mathrm{lin}}(\Sigma_1'), \beta \in \mathscr{T}(A_1'),$$
$$\mathscr{L}_1(N) = \mathscr{L}(c), \mathscr{L}_1(\beta) = \mathscr{L}(\tau(c)),$$
$$\vdash_{\Sigma_1'} N : \beta \,\},$$

$$\tau_0'(d_{\langle c, N, \beta \rangle}) = \mathrm{anti}(\tau(c), \beta),$$

where

$$\mathrm{anti}(p, \beta) = p^\beta,$$

$$\mathrm{anti}(\alpha_1 \to \alpha_2, \beta_1 \to \beta_2) = \mathrm{anti}(\alpha_1, \beta_1) \to \mathrm{anti}(\alpha_2, \beta_2).$$

Note that $\tau_0'(d_{\langle c, N, \beta \rangle})$ is always defined and is a most specific common anti-instance of $\tau(c)$ and β.

Construction

$\mathscr{L}' = \langle \sigma', \theta' \rangle$ is a lexicon from Σ'_0 to Σ'_1:

$$\sigma'(p^\beta) = \beta, \qquad\qquad \theta'(d_{\langle c,N,\beta \rangle}) = N.$$

Construction

$\mathscr{L}' = \langle \sigma', \theta' \rangle$ is a lexicon from Σ'_0 to Σ'_1:

$$\sigma'(p^\beta) = \beta, \qquad \theta'(d_{\langle c,N,\beta \rangle}) = N.$$

Define another lexicon $\mathscr{L}_0 = \langle \sigma_0, \theta_0 \rangle$ from Σ'_0 to Σ_0:

$$\sigma_0(p^\beta) = p, \qquad \theta_0(d_{\langle c,N,\beta \rangle}) = c.$$

Construction

$\mathscr{L}' = \langle \sigma', \theta' \rangle$ is a lexicon from Σ'_0 to Σ'_1:

$$\sigma'(p^{\beta}) = \beta, \qquad \theta'(d_{\langle c,N,\beta \rangle}) = N.$$

Define another lexicon $\mathscr{L}_0 = \langle \sigma_0, \theta_0 \rangle$ from Σ'_0 to Σ_0:

$$\sigma_0(p^{\beta}) = p, \qquad \theta_0(d_{\langle c,N,\beta \rangle}) = c.$$

We have $\mathscr{L} \circ \mathscr{L}_0 = \mathscr{L}_1 \circ \mathscr{L}'$:

$$\begin{array}{ccc}
\vdash_{\Sigma_0} c : \tau(c) & \xrightarrow{\mathscr{L}} & \vdash_{\Sigma_1} \mathscr{L}(c) : \mathscr{L}(\tau(c)) \\
\uparrow {\scriptstyle \mathscr{L}_0} & & \uparrow {\scriptstyle \mathscr{L}_1} \\
\vdash_{\Sigma'_0} d_{\langle c,N,\beta \rangle} : \mathrm{anti}(\tau(c), \beta) & \xrightarrow[\mathscr{L}']{} & \vdash_{\Sigma'_1} N : \beta
\end{array}$$

Proof of correctness

$$\mathcal{O}(\mathscr{G}_\cap) \subseteq \mathcal{O}(\mathscr{G}) \cap \{\, \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \,\}.$$

Proof of correctness

$$\mathcal{O}(\mathscr{G}_\cap) \subseteq \mathcal{O}(\mathscr{G}) \cap \{\, \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \gamma \,\}.$$

$$\mathscr{G}_\cap = \langle \Sigma_0', \Sigma_1, \mathscr{L}_1 \circ \mathscr{L}', s^\gamma \rangle$$

$$\begin{array}{ccc}
\vdash_{\Sigma_0} \mathscr{L}_0(P) : s & \xrightarrow{\mathscr{L}} & \vdash_{\Sigma_1} \mathscr{L}_1(|\mathscr{L}'(P)|_\beta) : \mathscr{L}_1(\gamma) \\
\mathscr{L}_0 \big\uparrow & & \big\uparrow \mathscr{L}_1 \\
\vdash_{\Sigma_0'} P : s^\gamma & \xrightarrow[\mathscr{L}']{} & \vdash_{\Sigma_1'} |\mathscr{L}'(P)|_\beta : \gamma
\end{array}$$

Proof of correctness

$$\mathcal{O}(\mathscr{G}_\cap) \subseteq \mathcal{O}(\mathscr{G}) \cap \{\, \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \gamma \,\}.$$

$$\mathscr{G}_\cap = \langle \Sigma_0', \Sigma_1, \mathscr{L}_1 \circ \mathscr{L}', s^\gamma \rangle$$

$$\begin{array}{ccc}
\vdash_{\Sigma_0} \mathscr{L}_0(P) : s & \xrightarrow{\mathscr{L}} & \vdash_{\Sigma_1} |\mathscr{L}(\mathscr{L}_0(P))|_\beta : \mathscr{L}(s) \\
\mathscr{L}_0 \Big\uparrow & & \Big\uparrow \mathscr{L}_1 \\
\vdash_{\Sigma_0'} P : s^\gamma & \xrightarrow[\mathscr{L}']{} & \vdash_{\Sigma_1'} |\mathscr{L}'(P)|_\beta : \gamma
\end{array}$$

Proof of correctness

$$\mathcal{O}(\mathscr{G}) \cap \{ \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \} \subseteq \mathcal{O}(\mathscr{G}_\cap).$$

Proof of correctness

$$\mathcal{O}(\mathscr{G}) \cap \{ \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \} \subseteq \mathcal{O}(\mathscr{G}_\cap).$$

Lemma. If $\mathscr{L}$ is a relabeling and $M \twoheadrightarrow_\beta \mathscr{L}(N)$ by non-erasing and non-duplicating β-reduction, then there is a P such that

$$\begin{array}{ccc} M & \twoheadrightarrow_\beta & \mathscr{L}(N) \\ \mathscr{L} \uparrow & & \mathscr{L} \uparrow \\ P & \twoheadrightarrow_\beta & N \end{array}$$

Proof of correctness

$$\mathcal{O}(\mathscr{G}) \cap \{\, \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \,\} \subseteq \mathcal{O}(\mathscr{G}_\cap).$$

$$\vdash_{\Sigma_0} P : s \quad \xrightarrow{\mathscr{L}} \quad \vdash_{\Sigma_1} \mathscr{L}(P) : \mathscr{L}(s) \quad \twoheadrightarrow_\beta \; \vdash_{\Sigma_1} \mathscr{L}_1(M) : \mathscr{L}(s)$$

$$\uparrow \mathscr{L}_1$$

$$\vdash_{\Sigma_1'} M : \boldsymbol{\gamma}$$

Proof of correctness

$$\mathcal{O}(\mathscr{G}) \cap \{\, \mathscr{L}_1(M) \mid \vdash_{\Sigma'_1} M : \boldsymbol{\gamma} \,\} \subseteq \mathcal{O}(\mathscr{G}_\cap).$$

$$\begin{array}{ccccc} \vdash_{\Sigma_0} P : s & \xrightarrow{\mathscr{L}} & \vdash_{\Sigma_1} \mathscr{L}(P) : \mathscr{L}(s) & \twoheadrightarrow_\beta & \vdash_{\Sigma_1} \mathscr{L}_1(M) : \mathscr{L}(s) \\ & & & & \uparrow \mathscr{L}_1 \\ & & & & \vdash_{\Sigma'_1} M : \boldsymbol{\gamma} \end{array}$$

$$\overrightarrow{Con}(P) = c_1 \ldots c_m$$

Proof of correctness

$$\mathcal{O}(\mathscr{G}) \cap \{\, \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \,\} \subseteq \mathcal{O}(\mathscr{G}_\cap).$$

$$\vdash_{\Sigma_0} \hat{P}[c_1, \ldots, c_m] : s \xrightarrow{\mathscr{L}} \vdash_{\Sigma_1} \hat{P}[\mathscr{L}(c_1), \ldots, \mathscr{L}(c_m)] : \mathscr{L}(s) \twoheadrightarrow_\beta \vdash_{\Sigma_1} \mathscr{L}_1(M) : \mathscr{L}(s)$$

$$\vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \xrightarrow{\mathscr{L}_1} \vdash_{\Sigma_1} \mathscr{L}_1(M) : \mathscr{L}(s)$$

$$\overrightarrow{Con}(P) = c_1 \ldots c_m$$

Proof of correctness

$$\mathcal{O}(\mathscr{G}) \cap \{\, \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \,\} \subseteq \mathcal{O}(\mathscr{G}_\cap).$$

$$\vdash_{\Sigma_0} \hat{P}[c_1, \ldots, c_m] : s \xrightarrow{\mathscr{L}} \vdash_{\Sigma_1} \hat{P}[\mathscr{L}(c_1), \ldots, \mathscr{L}(c_m)] : \mathscr{L}(s) \twoheadrightarrow_\beta \vdash_{\Sigma_1} \mathscr{L}_1(M) : \mathscr{L}(s)$$

$$\uparrow \mathscr{L}_1 \qquad\qquad \uparrow \mathscr{L}_1$$

$$\twoheadrightarrow_\beta \qquad \vdash_{\Sigma_1'} M : \boldsymbol{\gamma}$$

$$\overrightarrow{Con}(P) = c_1 \ldots c_m$$

Proof of correctness

$$\mathcal{O}(\mathscr{G}) \cap \{\, \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \,\} \subseteq \mathcal{O}(\mathscr{G}_\cap).$$

$$\begin{array}{ccccc}
\vdash_{\Sigma_0} \hat{P}[c_1, \ldots, c_m] : s & \xrightarrow{\mathscr{L}} & \vdash_{\Sigma_1} \hat{P}[\mathscr{L}(c_1), \ldots, \mathscr{L}(c_m)] : \mathscr{L}(s) & \twoheadrightarrow_\beta & \vdash_{\Sigma_1} \mathscr{L}_1(M) : \mathscr{L}(s) \\
 & & \uparrow \mathscr{L}_1 & & \uparrow \mathscr{L}_1 \\
 & & \hat{P}[N_1, \ldots, N_m] & \twoheadrightarrow_\beta & \vdash_{\Sigma_1'} M : \boldsymbol{\gamma}
\end{array}$$

$$\overrightarrow{Con}(P) = c_1 \ldots c_m$$

$$\mathscr{L}_1(N_i) = \mathscr{L}(c_i)$$

Proof of correctness

$$\mathcal{O}(\mathscr{G}) \cap \{ \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \} \subseteq \mathcal{O}(\mathscr{G}_\cap).$$

$$\begin{array}{ccccc}
\vdash_{\Sigma_0} \hat{P}[c_1, \ldots, c_m] : s & \xrightarrow{\mathscr{L}} & \vdash_{\Sigma_1} \hat{P}[\mathscr{L}(c_1), \ldots, \mathscr{L}(c_m)] : \mathscr{L}(s) & \twoheadrightarrow_\beta & \vdash_{\Sigma_1} \mathscr{L}_1(M) : \mathscr{L}(s) \\
 & & \uparrow \mathscr{L}_1 & & \uparrow \mathscr{L}_1 \\
 & & \vdash_{\Sigma_1'} \hat{P}[N_1, \ldots, N_m] : \boldsymbol{\gamma} & \twoheadrightarrow_\beta & \vdash_{\Sigma_1'} M : \boldsymbol{\gamma}
\end{array}$$

$$\overrightarrow{Con}(P) = c_1 \ldots c_m$$
$$\mathscr{L}_1(N_i) = \mathscr{L}(c_i)$$

Proof of correctness

$$\mathcal{O}(\mathcal{G}) \cap \{\, \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \,\} \subseteq \mathcal{O}(\mathcal{G}_\cap).$$

$$\begin{array}{ccccc}
\vdash_{\Sigma_0} \hat{P}[c_1, \ldots, c_m] : s & \xrightarrow{\mathscr{L}} & \vdash_{\Sigma_1} \hat{P}[\mathscr{L}(c_1), \ldots, \mathscr{L}(c_m)] : \mathscr{L}(s) & \twoheadrightarrow_\beta & \vdash_{\Sigma_1} \mathscr{L}_1(M) : \mathscr{L}(s) \\
 & & \uparrow \mathscr{L}_1 & & \uparrow \mathscr{L}_1 \\
 & & \vdash_{\Sigma_1'} \hat{P}[N_1, \ldots, N_m] : \boldsymbol{\gamma} & \twoheadrightarrow_\beta & \vdash_{\Sigma_1'} M : \boldsymbol{\gamma}
\end{array}$$

$$\overrightarrow{Con}(P) = c_1 \ldots c_m$$

$$\mathscr{L}_1(N_i) = \mathscr{L}(c_i)$$

$$\vdash_{\Sigma_1'} N_i : \beta_i$$

Proof of correctness

$$\mathcal{O}(\mathscr{G}) \cap \{\, \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \,\} \subseteq \mathcal{O}(\mathscr{G}_\cap).$$

$$\begin{array}{ccccc}
\vdash_{\Sigma_0} \hat{P}[c_1, \ldots, c_m] : s & \xrightarrow{\mathscr{L}} & \vdash_{\Sigma_1} \hat{P}[\mathscr{L}(c_1), \ldots, \mathscr{L}(c_m)] : \mathscr{L}(s) & \twoheadrightarrow_\beta & \vdash_{\Sigma_1} \mathscr{L}_1(M) : \mathscr{L}(s) \\
 & & \uparrow \mathscr{L}_1 & & \uparrow \mathscr{L}_1 \\
 & & \vdash_{\Sigma_1'} \hat{P}[N_1, \ldots, N_m] : \boldsymbol{\gamma} & \twoheadrightarrow_\beta & \vdash_{\Sigma_1'} M : \boldsymbol{\gamma}
\end{array}$$

$$\overrightarrow{Con}(P) = c_1 \ldots c_m$$

$$\mathscr{L}_1(N_i) = \mathscr{L}(c_i)$$

$$\vdash_{\Sigma_1'} N_i : \beta_i$$

$$\mathscr{L}_1(\beta_i) = \mathscr{L}(\tau_0(c_i))$$

Proof of correctness

$$\mathcal{O}(\mathscr{G}) \cap \{\, \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \,\} \subseteq \mathcal{O}(\mathscr{G}_\cap).$$

$$\begin{array}{ccccc}
\vdash_{\Sigma_0} \hat{P}[c_1, \ldots, c_m] : s & \xrightarrow{\mathscr{L}} & \vdash_{\Sigma_1} \hat{P}[\mathscr{L}(c_1), \ldots, \mathscr{L}(c_m)] : \mathscr{L}(s) & \twoheadrightarrow_\beta & \vdash_{\Sigma_1} \mathscr{L}_1(M) : \mathscr{L}(s) \\
 & & \uparrow \mathscr{L}_1 & & \uparrow \mathscr{L}_1 \\
 & & \vdash_{\Sigma_1'} \hat{P}[N_1, \ldots, N_m] : \boldsymbol{\gamma} & \twoheadrightarrow_\beta & \vdash_{\Sigma_1'} M : \boldsymbol{\gamma}
\end{array}$$

$$\begin{aligned}
\overrightarrow{Con}(P) &= c_1 \ldots c_m \\
\mathscr{L}_1(N_i) &= \mathscr{L}(c_i) \\
&\vdash_{\Sigma_1'} N_i : \beta_i \\
\mathscr{L}_1(\beta_i) &= \mathscr{L}(\tau_0(c_i)) \\
d_{\langle c_i, N_i, \beta_i \rangle} &\in A_0' \\
\tau_0'(d_{\langle c_i, N_i, \beta_i \rangle}) &= \mathrm{anti}(\tau_0(c_i), \beta_i)
\end{aligned}$$

Proof of correctness

$$\mathcal{O}(\mathscr{G}) \cap \{\, \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \,\} \subseteq \mathcal{O}(\mathscr{G}_\cap).$$

$$\begin{array}{ccccc}
\vdash_{\Sigma_0} \hat{P}[c_1, \dots, c_m] : s & \xrightarrow{\mathscr{L}} & \vdash_{\Sigma_1} \hat{P}[\mathscr{L}(c_1), \dots, \mathscr{L}(c_m)] : \mathscr{L}(s) & \twoheadrightarrow_\beta & \vdash_{\Sigma_1} \mathscr{L}_1(M) : \mathscr{L}(s) \\
 & & \uparrow \mathscr{L}_1 & & \uparrow \mathscr{L}_1 \\
 & & \vdash_{\Sigma_1'} \hat{P}[N_1, \dots, N_m] : \boldsymbol{\gamma} & \twoheadrightarrow_\beta & \vdash_{\Sigma_1'} M : \boldsymbol{\gamma}
\end{array}$$

$$\begin{aligned}
\overrightarrow{Con}(P) &= c_1 \dots c_m \\
\mathscr{L}_1(N_i) &= \mathscr{L}(c_i) \\
&\vdash_{\Sigma_1'} N_i : \beta_i \\
\mathscr{L}_1(\beta_i) &= \mathscr{L}(\tau_0(c_i)) \\
d_{\langle c_i, N_i, \beta_i \rangle} &\in A_0' \\
\tau_0'(d_{\langle c_i, N_i, \beta_i \rangle}) &= \mathrm{anti}(\tau_0(c_i), \beta_i)
\end{aligned}$$

$$y_1 : \tau_0(c_1), \dots, y_m : \tau_0(c_m) \vdash \hat{P}[y_1, \dots, y_m] : s$$

$$y_1 : \beta_1, \dots, y_m : \beta_m \vdash \hat{P}[y_1, \dots, y_m] : \boldsymbol{\gamma}$$

Proof of correctness

$$\mathcal{O}(\mathscr{G}) \cap \{\, \mathscr{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \,\} \subseteq \mathcal{O}(\mathscr{G}_\cap).$$

$$\begin{array}{ccccc}
\vdash_{\Sigma_0} \hat{P}[c_1, \ldots, c_m] : s & \xrightarrow{\mathscr{L}} & \vdash_{\Sigma_1} \hat{P}[\mathscr{L}(c_1), \ldots, \mathscr{L}(c_m)] : \mathscr{L}(s) & \twoheadrightarrow_\beta & \vdash_{\Sigma_1} \mathscr{L}_1(M) : \mathscr{L}(s) \\
 & & \uparrow \mathscr{L}_1 & & \uparrow \mathscr{L}_1 \\
 & & \vdash_{\Sigma_1'} \hat{P}[N_1, \ldots, N_m] : \boldsymbol{\gamma} & \twoheadrightarrow_\beta & \vdash_{\Sigma_1'} M : \boldsymbol{\gamma}
\end{array}$$

$$\begin{aligned}
\overrightarrow{Con}(P) &= c_1 \ldots c_m \\
\mathscr{L}_1(N_i) &= \mathscr{L}(c_i) \\
&\vdash_{\Sigma_1'} N_i : \beta_i \\
\mathscr{L}_1(\beta_i) &= \mathscr{L}(\tau_0(c_i)) \\
d_{\langle c_i, N_i, \beta_i \rangle} &\in A_0' \\
\tau_0'(d_{\langle c_i, N_i, \beta_i \rangle}) &= \mathrm{anti}(\tau_0(c_i), \beta_i)
\end{aligned}$$

$$\begin{aligned}
y_1 : \tau_0(c_1), \ldots, y_m : \tau_0(c_m) &\vdash \hat{P}[y_1, \ldots, y_m] : s \\
y_1 : \beta_1, \ldots, y_m : \beta_m &\vdash \hat{P}[y_1, \ldots, y_m] : \boldsymbol{\gamma} \\
y_1 : \mathrm{anti}(\tau_0(c_1), \beta_1), \ldots, y_m : \mathrm{anti}(\tau_0(c_m), \beta_m) &\vdash \hat{P}[y_1, \ldots, y_m] : s^{\boldsymbol{\gamma}}
\end{aligned}$$

Proof of correctness

$$\mathcal{O}(\mathcal{G}) \cap \{\, \mathcal{L}_1(M) \mid \vdash_{\Sigma_1'} M : \boldsymbol{\gamma} \,\} \subseteq \mathcal{O}(\mathcal{G}_\cap).$$

$$\begin{array}{ccccc}
\vdash_{\Sigma_0} \hat{P}[c_1, \ldots, c_m] : s & \xrightarrow{\mathcal{L}} & \vdash_{\Sigma_1} \hat{P}[\mathcal{L}(c_1), \ldots, \mathcal{L}(c_m)] : \mathcal{L}(s) & \twoheadrightarrow_\beta & \vdash_{\Sigma_1} \mathcal{L}_1(M) : \mathcal{L}(s) \\
\uparrow \mathcal{L}_0 & & \uparrow \mathcal{L}_1 & & \uparrow \mathcal{L}_1 \\
\vdash_{\Sigma_0'} \hat{P}[d_{\langle c_1, N_1, \beta_1 \rangle}, \ldots, d_{\langle c_m, N_m, \beta_m \rangle}] : s^{\gamma} & \xrightarrow[\mathcal{L}']{} & \vdash_{\Sigma_1'} \hat{P}[N_1, \ldots, N_m] : \boldsymbol{\gamma} & \twoheadrightarrow_\beta & \vdash_{\Sigma_1'} M : \boldsymbol{\gamma}
\end{array}$$

$$\begin{aligned}
\overrightarrow{Con}(P) &= c_1 \ldots c_m \\
\mathcal{L}_1(N_i) &= \mathcal{L}(c_i) \\
&\vdash_{\Sigma_1'} N_i : \beta_i \\
\mathcal{L}_1(\beta_i) &= \mathcal{L}(\tau_0(c_i)) \\
d_{\langle c_i, N_i, \beta_i \rangle} &\in A_0' \\
\tau_0'(d_{\langle c_i, N_i, \beta_i \rangle}) &= \mathrm{anti}(\tau_0(c_i), \beta_i)
\end{aligned}$$

$$\begin{aligned}
y_1 : \tau_0(c_1), \ldots, y_m : \tau_0(c_m) &\vdash \hat{P}[y_1, \ldots, y_m] : s \\
y_1 : \beta_1, \ldots, y_m : \beta_m &\vdash \hat{P}[y_1, \ldots, y_m] : \boldsymbol{\gamma} \\
y_1 : \mathrm{anti}(\tau_0(c_1), \beta_1), \ldots, y_m : \mathrm{anti}(\tau_0(c_m), \beta_m) &\vdash \hat{P}[y_1, \ldots, y_m] : s^{\gamma}
\end{aligned}$$

Application: Parsing as intersection

Theorem. UNIVERSAL RECOGNITION reduces to NON-EMPTINESS.

$$
\begin{aligned}
M \in \mathcal{O}(\mathscr{G}) &\iff \mathcal{O}(\mathscr{G}) \cap \{M\} \neq \varnothing \\
&\iff \mathcal{O}(\mathscr{G}_\cap) \neq \varnothing
\end{aligned}
$$

Application: Parsing as intersection

Lemma. A singleton set is the image of a relabeling.

Application: Parsing as intersection

Lemma. A singleton set is the image of a relabeling.

Take $M \in \Lambda_{\text{lin}}(\Sigma)$ in long normal form with

$$\vdash_\Sigma M : \beta$$

Let $\overrightarrow{Con}(M) = a_1 \ldots a_n$, and let $\hat{M}[x_1, \ldots, x_n] \in \Lambda_{\text{lin}}$ be such that $M = \hat{M}[a_1, \ldots, a_n]$.

Application: Parsing as intersection

Lemma. A singleton set is the image of a relabeling.

Take $M \in \Lambda_{\mathrm{lin}}(\Sigma)$ in long normal form with

$$\vdash_\Sigma M : \beta$$

Let $\overrightarrow{Con}(M) = a_1 \ldots a_n$, and let $\hat{M}[x_1, \ldots, x_n] \in \Lambda_{\mathrm{lin}}$ be such that $M = \hat{M}[a_1, \ldots, a_n]$.

Let

$$x_1 : \alpha_1, \ldots, x_n : \alpha_n \vdash \hat{M}[x_1, \ldots, x_n] : \alpha$$

be a principal pair for $\hat{M}[x_1, \ldots, x_n]$. Since $\hat{M}[x_1, \ldots, x_n]$ is linear, $\alpha_1, \ldots, \alpha_n \vdash \alpha$ is a balanced sequent.

Application: Parsing as intersection

Lemma. A singleton set is the image of a relabeling.

Take $M \in \Lambda_{\text{lin}}(\Sigma)$ in long normal form with

$$\vdash_{\Sigma} M : \beta$$

Let $\overrightarrow{Con}(M) = a_1 \ldots a_n$, and let $\hat{M}[x_1, \ldots, x_n] \in \Lambda_{\text{lin}}$ be such that $M = \hat{M}[a_1, \ldots, a_n]$.

Let

$$x_1 : \alpha_1, \ldots, x_n : \alpha_n \vdash \hat{M}[x_1, \ldots, x_n] : \alpha$$

be a principal pair for $\hat{M}[x_1, \ldots, x_n]$. Since $\hat{M}[x_1, \ldots, x_n]$ is linear, $\alpha_1, \ldots, \alpha_n \vdash \alpha$ is a balanced sequent.

By the Coherence Theorem,

$$\Gamma \vdash N : \alpha \quad \text{for some } \Gamma \subseteq \{x_1 : \alpha_1, \ldots, x_n : \alpha_n\}$$

implies $N =_{\beta\eta} \hat{M}[x_1, \ldots, x_n]$.

Application: Parsing as intersection

Define $\Sigma' = \langle A', C', \tau' \rangle$:

$$A' = \text{the set of atomic types in } \alpha_1, \ldots, \alpha_n, \alpha,$$

$$C' = \{a'_1, \ldots, a'_n\}, \quad \textit{distinct fresh constants}$$

$$\tau'(a'_i) = \alpha_i.$$

Application: Parsing as intersection

Define $\Sigma' = \langle A', C', \tau' \rangle$:

$$
\begin{aligned}
A' &= \text{the set of atomic types in } \alpha_1, \ldots, \alpha_n, \alpha, \\
C' &= \{a'_1, \ldots, a'_n\}, \quad \textit{distinct fresh constants} \\
\tau'(a'_i) &= \alpha_i.
\end{aligned}
$$

Define a relabeling $\mathscr{L} = \langle \sigma, \theta \rangle$ from Σ' to Σ:

$$
\begin{aligned}
\sigma &\text{ is such that } \sigma(\alpha_i) = \tau(a_i), \sigma(\alpha) = \beta, \\
\theta(a'_i) &= a_i.
\end{aligned}
$$

Application: Parsing as intersection

Define $\Sigma' = \langle A', C', \tau' \rangle$:

$$A' = \text{the set of atomic types in } \alpha_1, \ldots, \alpha_n, \alpha,$$

$$C' = \{a'_1, \ldots, a'_n\}, \quad \textit{distinct fresh constants}$$

$$\tau'(a'_i) = \alpha_i.$$

Define a relabeling $\mathscr{L} = \langle \sigma, \theta \rangle$ from Σ' to Σ:

$$\sigma \text{ is such that } \sigma(\alpha_i) = \tau(a_i), \sigma(\alpha) = \beta,$$

$$\theta(a'_i) = a_i.$$

$\vdash_{\Sigma'} N : \alpha$ implies $N =_{\beta\eta} \hat{M}[a'_1, \ldots, a'_n]$.

Application: Parsing as intersection

Define $\Sigma' = \langle A', C', \tau' \rangle$:

$$A' = \text{the set of atomic types in } \alpha_1, \ldots, \alpha_n, \alpha,$$
$$C' = \{a'_1, \ldots, a'_n\}, \quad \textit{distinct fresh constants}$$
$$\tau'(a'_i) = \alpha_i.$$

Define a relabeling $\mathscr{L} = \langle \sigma, \theta \rangle$ from Σ' to Σ:

$$\sigma \text{ is such that } \sigma(\alpha_i) = \tau(a_i), \sigma(\alpha) = \beta,$$
$$\theta(a'_i) = a_i.$$

$\vdash_{\Sigma'} N : \alpha$ implies $N =_{\beta\eta} \hat{M}[a'_1, \ldots, a'_n]$.

Gives a quick proof that second-order ACGs generate PTIME languages (Salvati 2005).