Generation with Grammars enriched with Lexical Semantics Information

Pierre Bourreau[1] Sylvain Salvati[1]

[1]Equipe SIGNES
LaBRI - INRIA Sud-Ouest

Introduction

- Goal: extend parsing techniques on ACG by adding new operation (here: deletion)
  - Parsing ACG $\Rightarrow$ Natural Language Generation
- Deletion can be used to represent lexical semantics information in our grammar
  - No intension of creating a new lexical semantics theory.

Outline

Second-order ACG and Lexical Semantics
- Abstract Categorial Grammars
- Integrating some lexical semantics information

Parsing ACG
- General Idea
- Using types

Extended parsers
- Typing issues
- A new typing system
- Example and Datalog

Outline

Second-order ACG and Lexical Semantics
- Abstract Categorial Grammars
- Integrating some lexical semantics information

Parsing ACG
- General Idea
- Using types

Extended parsers
- Typing issues
- A new typing system
- Example and Datalog

ACG

- [dG01, Mus01]
- Computational linguistics.
- Focus on syntax, semantics and their relation.
- Based on two main ideas:
  - Montagovian semantics,
  - Curry's distinction between phenogrammar and tectogrammar.

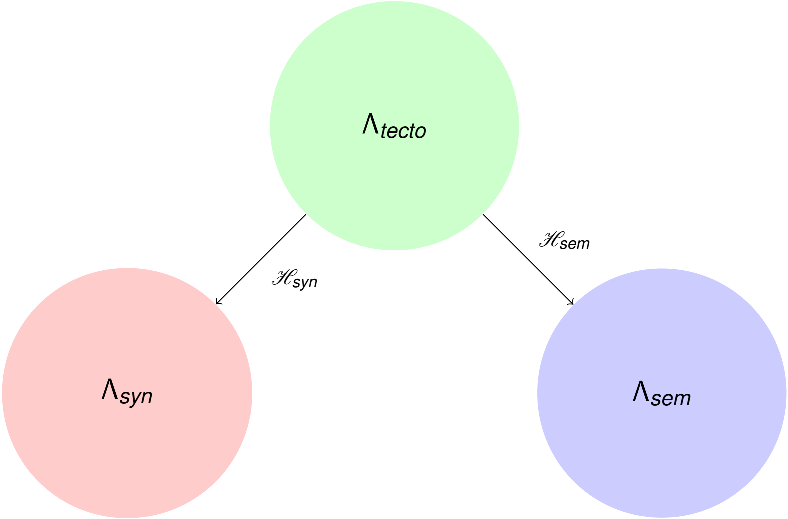
ACG

- [dG01, Mus01]
- Computational linguistics.
- Focus on syntax, semantics and their relation.
- Based on two main ideas:
  - Montagovian semantics, λ-calculus for semantics
  - Curry's distinction between phenogrammar and tectogrammar. intermediate structure between syntax and semantics

ACG

- [dG01, Mus01]
- Computational linguistics.
- Focus on syntax, semantics and their relation.
- Based on two main ideas:
  - Montagovian semantics, λ-calculus for semantics
  - Curry's distinction between phenogrammar and tectogrammar. intermediate structure between syntax and semantics
- Plus, uniformity of the formalism: use of the λ-calculus to describe every module/grammar

Example


Λ_{tecto}
$\mathcal{H}_{syn}$
$\mathcal{H}_{sem}$
Λ_{syn}
Λ_{sem}

From tectogrammars to phenogrammars

The lexicons

- We use homomorphisms.
- Nothing new:
  - [Mon73], [Lam58]
- If terms are typed, $\mathscr{H}$ applies to both terms and types.

As an example (syntax)

- *eat*: $\mathtt{np} \rightarrow \mathtt{np} \rightarrow \mathtt{s}$
  - $\mathscr{H}_{syn}(\mathtt{np}) = \mathtt{str}$
  - $\mathscr{H}_{syn}(\mathtt{s}) = \mathtt{str}$
  - $\mathscr{H}_{syn}(\lambda xy.eat\,xy) = \lambda x_1 x_2 . x_2 + \mathbf{eat} + x_1$

From tectogrammars to phenogrammars

The lexicons

- We use homomorphisms.
- Nothing new:
  - [Mon73], [Lam58].
- If terms are typed, $\mathscr{H}$ applies to both terms and types.

As an example (semantics)

- $\mathit{eat}{:}\mathtt{np} \rightarrow \mathtt{np} \rightarrow \mathtt{s}$
  - $\mathscr{H}_{sem}(\mathtt{np}) = (\mathtt{e} \rightarrow \mathtt{t}) \rightarrow \mathtt{t}$
  - $\mathscr{H}_{sem}(\mathtt{s}) = \mathtt{t}$
  - $\mathscr{H}_{sem}(\lambda xy.\mathit{eat}\,xy) = \lambda PQ.P(\lambda x.Q(\lambda y.\mathit{EAT}\,xy))$

Formally

Higher-Order Signature

A higher-order signature $\Sigma = (\mathscr{A}, \mathscr{C}, \tau)$:

- $\mathscr{A}$ a finite set of atomic types
- $\mathscr{C}$ a finite set of constants
- τ the typing function $\mathscr{C} \to \mathscr{T}(\mathscr{A})$

Derivation system

$$\frac{}{x : \alpha \vdash_\Sigma x : \alpha} \qquad \frac{}{\vdash_\Sigma \boldsymbol{c} : \tau(\boldsymbol{c})}$$

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

Overview (1)

- An ACG $\mathscr{G} = (\Sigma_1, \Sigma_2, \mathscr{H}, s)$
  - $\mathscr{A}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_1} | \vdash_{\Sigma_1} M : s\}$
  - $\mathscr{O}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_2} | \exists N \in \mathscr{A}(\mathscr{G}), |\mathscr{H}(N)|_\beta = M\}$
- Terms of the tectogrammar represent the deep structure of a sentence.
- Syntax is a realization of this structure...
- Just like semantics!
- λ-terms used to represent all this structures.

Overview (1)

- An ACG $\mathscr{G} = (\Sigma_1, \Sigma_2, \mathscr{H}, s)$
  - $\mathscr{A}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_1} | \vdash_{\Sigma_1} M : s\}$
  - $\mathscr{O}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_2} | \exists N \in \mathscr{A}(\mathscr{G}), |\mathscr{H}(N)|_\beta = M\}$
- Terms of the tectogrammar represent the deep structure of a sentence.
- Syntax is a realization of this structure...
- Just like semantics!
- λ-terms used to represent all this structures.

Overview (1)

- An ACG $\mathscr{G} = (\Sigma_1, \Sigma_2, \mathscr{H}, s)$
  - $\mathscr{A}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_1} | \vdash_{\Sigma_1} M : s\}$
  - $\mathscr{O}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_2} | \exists N \in \mathscr{A}(\mathscr{G}), |\mathscr{H}(N)|_\beta = M\}$
- Terms of the tectogrammar represent the deep structure of a sentence.
- Syntax is a realization of this structure...
- Just like semantics!
- λ-terms used to represent all this structures.

Overview (1)

- An ACG $\mathscr{G} = (\Sigma_1, \Sigma_2, \mathscr{H}, s)$
  - $\mathscr{A}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_1} | \vdash_{\Sigma_1} M : s\}$
  - $\mathscr{O}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_2} | \exists N \in \mathscr{A}(\mathscr{G}), |\mathscr{H}(N)|_\beta = M\}$
- Terms of the tectogrammar represent the deep structure of a sentence.
- Syntax is a realization of this structure...
- Just like semantics!
- λ-terms used to represent all this structures.

Overview (1)

- An ACG $\mathscr{G} = (\Sigma_1, \Sigma_2, \mathscr{H}, s)$
  - $\mathscr{A}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_1} | \vdash_{\Sigma_1} M : s\}$
  - $\mathscr{O}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_2} | \exists N \in \mathscr{A}(\mathscr{G}), |\mathscr{H}(N)|_\beta = M\}$
- Terms of the tectogrammar represent the deep structure of a sentence.
- Syntax is a realization of this structure...
- Just like semantics!
- λ-terms used to represent all this structures.

Overview (1)

- An ACG $\mathscr{G} = (\Sigma_1, \Sigma_2, \mathscr{H}, s)$
  - $\mathscr{A}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_1} | \vdash_{\Sigma_1} M : s\}$
  - $\mathscr{O}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_2} | \exists N \in \mathscr{A}(\mathscr{G}), |\mathscr{H}(N)|_\beta = M\}$
- Terms of the tectogrammar represent the deep structure of a sentence.
- Syntax is a realization of this structure...
- Just like semantics!
- λ-terms used to represent all this structures.

Overview (1)

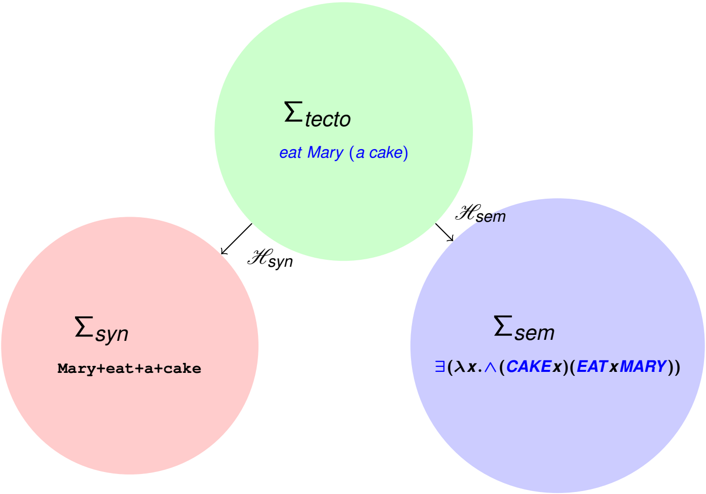
- An ACG $\mathscr{G} = (\Sigma_1, \Sigma_2, \mathscr{H}, s)$
  - $\mathscr{A}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_1} | \vdash_{\Sigma_1} M : s\}$
  - $\mathscr{O}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_2} | \exists N \in \mathscr{A}(\mathscr{G}), |\mathscr{H}(N)|_\beta = M\}$
- Terms of the tectogrammar represent the deep structure of a sentence.
- Syntax is a realization of this structure...
- Just like semantics!
- λ-terms used to represent all this structures.

Overview (1)

- An ACG $\mathscr{G} = (\Sigma_1, \Sigma_2, \mathscr{H}, s)$
  - $\mathscr{A}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_1} | \vdash_{\Sigma_1} M : s\}$
  - $\mathscr{O}(\mathscr{G}) = \{M \in \Lambda_{\Sigma_2} | \exists N \in \mathscr{A}(\mathscr{G}), |\mathscr{H}(N)|_\beta = M\}$
- Terms of the tectogrammar represent the deep structure of a sentence.
- Syntax is a realization of this structure...
- Just like semantics!
- λ-terms used to represent all this structures.

NL Generation $\equiv$ NL Parsing

Overview(2)


Σ_{tecto}
eat Mary (a cake)
$\mathscr{H}_{syn}$
$\mathscr{H}_{sem}$
Σ_{syn}
Mary+eat+a+cake
Σ_{sem}
$\exists(\lambda x.\wedge(CAKE\, x)(EAT\, x\, MARY))$

Outline

Second-order ACG and Lexical Semantics
- Abstract Categorial Grammars
- Integrating some lexical semantics information

Parsing ACG
- General Idea
- Using types

Extended parsers
- Typing issues
- A new typing system
- Example and Datalog

Original ACG

Linearity

A term M is linear if every variable in M has one and only one occurrence in M (no deletion, no copy)

Example

x, $\lambda x.fx$ but not $\lambda x.fxx$

Original ACG

Linearity

A term M is linear if every variable in M has one and only one occurrence in M (no deletion, no copy)

Example

x, $\lambda x.fx$ but not $\lambda x.fxx$

(Linear) ACG

$\mathscr{G} = (\Sigma_1, \Sigma_2, \mathscr{H}, s)$. For every constant c of Σ_1, $\mathscr{H}(c)$ is linear.

First extension

Almost Linearity

A term M is almost linear if every variable in M has at least one occurrence in M (no deletion).
A variable which has more than one occurrence in M is assigned an atomic type in M's principal typing limited copy)

Example

x, $\lambda x.fx$, $\lambda x.fxx$ but not $\lambda x.f(fx)$

First extension

Almost Linearity

A term M is almost linear if every variable in M has at least one occurrence in M (no deletion).
A variable which has more than one occurrence in M is assigned an atomic type in M's principal typing limited copy)

Example

x, $\lambda x.fx$, $\lambda x.fxx$ but not $\lambda x.f(fx)$

Almost linear ACG

$\mathscr{G} = (\Sigma_1, \Sigma_2, \mathscr{H}, s)$. For every constant c of Σ_1, $\mathscr{H}(c)$ is almost linear.

Lexical Semantics: what kind of information?

Aspects

- "John bought and read Hamlet".
- Hamlet: the character? A book as an object? A book as an information container?
- Semantics:
  - $\wedge (\mathit{BUY\ HAM\ JOHN})\ (\mathit{READ\ HAM\ JOHN})$
  - Differenciation through **terms** and not types (Pustejovsky)
    - $\wedge (\mathit{BUY\ HAM}_{phys-obj}\ \mathit{JOHN})\ (\mathit{READ\ HAM}_{info-cont}\ \mathit{JOHN})$
    - $\wedge (\mathit{BUY\ HAM}_{phys-obj}\ \mathit{JOHN}_{char})\ (\mathit{READ\ HAM}_{info-cont}\ \mathit{JOHN}_{char})$

Lexical Semantics: what kind of information?

Aspects

- `“John bought and read Hamlet”.`
- Hamlet: the character? A book as an object? A book as an information container?
- Semantics:
  - $\wedge(BUY\ HAM\ JOHN)\ (READ\ HAM\ JOHN)$
  - Differenciation through **terms** and not types (Pustejovsky)
    - $\wedge(BUY\ HAM_{phys-obj}\ JOHN)\ (READ\ HAM_{info-cont}\ JOHN)$
    - $\wedge(BUY\ HAM_{phys-obj}\ JOHN_{char})\ (READ\ HAM_{info-cont}\ JOHN_{char})$

Lexical Semantics: what kind of information?

Aspects

- “John bought and read Hamlet”.
- Hamlet: the character? A book as an object? A book as an information container?
- Semantics:
  - $\wedge(\mathit{BUY}\ \mathit{HAM}\ \mathit{JOHN})\ (\mathit{READ}\ \mathit{HAM}\ \mathit{JOHN})$
  - Differenciation through **terms** and not types (Pustejovsky)
    - $\wedge(\mathit{BUY}\ \mathit{HAM}_{phys-obj}\ \mathit{JOHN})\ (\mathit{READ}\ \mathit{HAM}_{info-cont}\ \mathit{JOHN})$
    - $$\wedge(\mathit{BUY}\ \mathit{HAM}_{phys-obj}\ \mathit{JOHN}_{char})\ (\mathit{READ}\ \mathit{HAM}_{info-cont}\ \mathit{JOHN}_{char})$$

Lexical Semantics: what kind of information?

Aspects

- `“John bought and read Hamlet”.`
- Hamlet: the character? A book as an object? A book as an information container?
- Semantics:
  - $\wedge (\mathit{BUY\ HAM\ JOHN})\ (\mathit{READ\ HAM\ JOHN})$
  - Differenciation through **terms** and not types (Pustejovsky)
    - $\wedge (\mathit{BUY\ HAM}_{phys-obj}\ \mathit{JOHN})\ (\mathit{READ\ HAM}_{info-cont}\ \mathit{JOHN})$
    - $\wedge (\mathit{BUY\ HAM}_{phys-obj}\ \mathit{JOHN}_{char})\ (\mathit{READ\ HAM}_{info-cont}\ \mathit{JOHN}_{char})$

Lexical Semantics: what kind of information?

Aspects

- "John bought and read Hamlet".
- Hamlet: the character? A book as an object? A book as an information container?
- Semantics:
  - $\wedge(BUY\ HAM\ JOHN)\ (READ\ HAM\ JOHN)$
  - Differenciation through **terms** and not types (Pustejovsky)
    - $\wedge(BUY\ HAM_{phys-obj}\ JOHN)\ (READ\ HAM_{info-cont}\ JOHN)$
    - $\wedge(BUY\ HAM_{phys-obj}\ JOHN_{char})\ (READ\ HAM_{info-cont}\ JOHN_{char})$

Lexical Semantics: what kind of information?

Aspects

- "John bought and read Hamlet".
- Hamlet: the character? A book as an object? A book as an information container?
- Semantics:
  - $\wedge (BUY\ HAM\ JOHN)\ (READ\ HAM\ JOHN)$
  - Differenciation through **terms** and not types (Pustejovsky)
    - $\wedge (BUY\ HAM_{phys-obj}\ JOHN)\ (READ\ HAM_{info-cont}\ JOHN)$
    - $\wedge (BUY\ HAM_{phys-obj}\ JOHN_{char})\ (READ\ HAM_{info-cont}\ JOHN_{char})$

Lexical Semantics: what kind of information?

Aspects

- "John bought and read Hamlet".
- Hamlet: the character? A book as an object? A book as an information container?
- Semantics:
  - $\wedge(BUY\ HAM\ JOHN)\ (READ\ HAM\ JOHN)$
  - Differenciation through **terms** and not types (Pustejovsky)
    - $\wedge(BUY\ HAM_{phys-obj}\ JOHN)\ (READ\ HAM_{info-cont}\ JOHN)$
    - $\wedge(BUY\ HAM_{phys-obj}\ JOHN_{char})\ (READ\ HAM_{info-cont}\ JOHN_{char})$

Choice as deletion

List of aspects on NP

- $\mathscr{H}_{sem}(\mathit{hamlet}) = \lambda P.P\ \mathit{HAM}$

Verb (predicate) as selector

- $\mathscr{H}_{sem}(\mathit{read}) = \lambda PQ.P(\lambda x.Q(\lambda y.\mathit{READ}xy))$

Choice as deletion

List of aspects on NP

- $\mathscr{H}_{syn}(hamlet) = \lambda QP.P(Q\ HAM_{char}\ HAM_{phys-obj}\ HAM_{info-cont})$
- Q is the selector

Verb (predicate) as selector

- $\mathscr{H}_{sem}(\text{read}) = \lambda PQ.P\pi_3(\lambda x.Q\pi_1(\lambda y.READ\,xy))$
- $\pi_i = \lambda x_1 x_2 x_3 . x_i$

Almost affine terms

Almost affine terms

A term M is almost affine if every variable/constant which has more than one occurrence in M is assigned an atomic type in M's principal typing

Example

$\lambda x^a y^b . f^{a \rightarrow a \rightarrow c} x^a x^a$ but not $\lambda x^a y^b . f^{a \rightarrow a \rightarrow a}(f^{a \rightarrow a \rightarrow a} x^a x^a) x^a$

Almost affine ACG

An ACG $(\Sigma_1, \Sigma_2, \mathscr{L}, s)$ is almost affine if for every constant c in Σ_1, $\mathscr{L}(c)$ is almost affine.

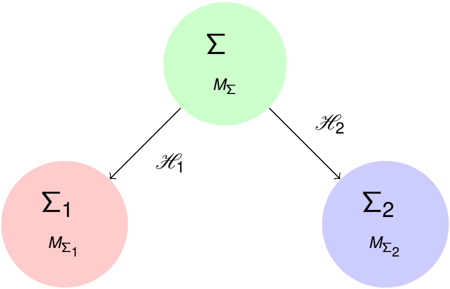
Outline

Second-order ACG and Lexical Semantics
- Abstract Categorial Grammars
- Integrating some lexical semantics information

Parsing ACG
- General Idea
- Using types

Extended parsers
- Typing issues
- A new typing system
- Example and Datalog

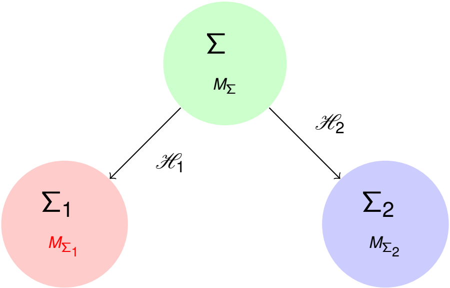
Parsing ACG



Sketch

1. A term $M_{\Sigma_1} : \alpha$ in Σ_1
2. Find the terms M_Σ, such that $\mathscr{H}_1(M_\Sigma) \twoheadrightarrow_\beta M_{\Sigma_1}$
3. Get the terms M_{Σ_2}, such that $\mathscr{H}_2(M_\Sigma) \twoheadrightarrow_\beta M_{\Sigma_2}$

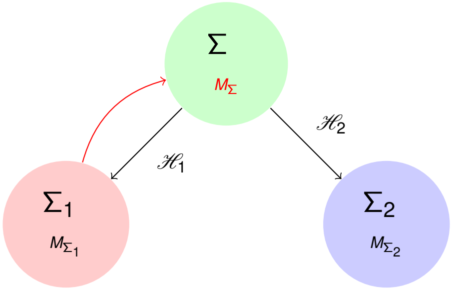
Parsing ACG



Sketch

1. A term $M_{\Sigma_1} : \alpha$ in Σ_1
2. Find the terms M_Σ, such that $\mathscr{H}_1(M_\Sigma) \twoheadrightarrow_\beta M_{\Sigma_1}$
3. Get the terms M_{Σ_2}, such that $\mathscr{H}_2(M_\Sigma) \twoheadrightarrow_\beta M_{\Sigma_2}$

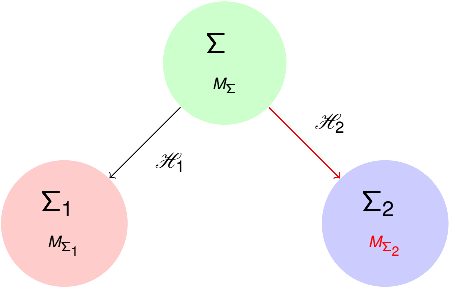
Parsing ACG



Sketch

1. A term $M_{\Sigma_1} : \alpha$ in Σ_1
2. Find the terms M_Σ, such that $\mathscr{H}_1(M_\Sigma) \twoheadrightarrow_\beta M_{\Sigma_1}$
3. Get the terms M_{Σ_2}, such that $\mathscr{H}_2(M_\Sigma) \twoheadrightarrow_\beta M_{\Sigma_2}$

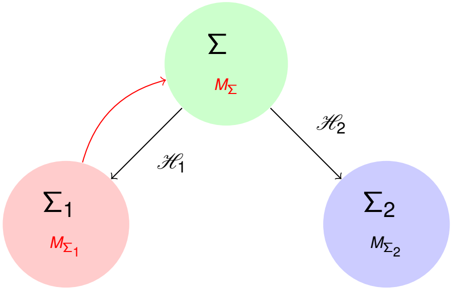
Parsing ACG



Sketch

1. A term $M_{\Sigma_1} : \alpha$ in Σ_1
2. Find the terms M_Σ, such that $\mathscr{H}_1(M_\Sigma) \twoheadrightarrow_\beta M_{\Sigma_1}$
3. Get the terms M_{Σ_2}, such that $\mathscr{H}_2(M_\Sigma) \twoheadrightarrow_\beta M_{\Sigma_2}$

Parsing ACG



Sketch

2. Find the term M_Σ, such that $\mathscr{H}_1(M_\Sigma) \twoheadrightarrow_\beta M_{\Sigma_1}$

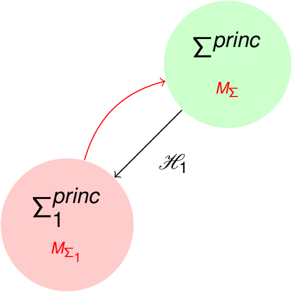
Outline

Second-order ACG and Lexical Semantics
- Abstract Categorial Grammars
- Integrating some lexical semantics information

Parsing ACG
- General Idea
- Using types

Extended parsers
- Typing issues
- A new typing system
- Example and Datalog

Idea: Use Types



If M_{Σ_1} and $\mathscr{H}_1(M_{\Sigma})$ share the **same principal typing** then $M_{\Sigma_1} =_{\beta} \mathscr{H}_1(M_{\Sigma})$

Idea: Use Types

Theorem

[Coherence] Let's consider a β-reduced term M and $\langle \Gamma ; \gamma \rangle$ its principal typing. If M is ??? it is the unique β-normal inhabitant of $\langle \Gamma ; \gamma \rangle$

Theorem

[Subject Expansion] Let's consider a ??? term M, a term M' such that $M \twoheadrightarrow_\beta M'$ and $\Gamma \vdash M' : \gamma$. Then $\Gamma \vdash M : \gamma$

Idea: Use Types

Theorem

[Coherence] Let's consider a β-reduced term M and $\langle \Gamma; \gamma \rangle$ its principal typing. If M is linear it is the unique β-normal inhabitant of $\langle \Gamma; \gamma \rangle$ [BS82]

Theorem

[Subject Expansion] Let's consider a linear term M, a term M' such that $M \twoheadrightarrow_\beta M'$ and $\Gamma \vdash M' : \gamma$. Then $\Gamma \vdash M : \gamma$

Idea: Use Types

Theorem

[Coherence] Let's consider a β-reduced term M and $\langle\Gamma;\gamma\rangle$ its principal typing. If M is almost linear it is the unique β-normal inhabitant of $\langle\Gamma;\gamma\rangle$ [Aot99]

Theorem

[Subject Expansion] Let's consider a almost linear term M, a term M' such that $M \twoheadrightarrow_\beta M'$ and $\Gamma \vdash M' : \gamma$. Then $\Gamma \vdash M : \gamma$ [Kan07]

Results

- [Kan07] gave a Datalog recognizer for **linear** and **almost linear** terms.
  - Complexity is **LOGCFL** $\subseteq$ **P**
- [Sal10] proved natural language generation is decidable in the Montagovian framework

Results

- [Kan07] gave a Datalog recognizer for **linear** and **almost linear** terms.
  - Complexity is **LOGCFL** $\subseteq$ **P**
- [Sal10] proved natural language generation is decidable in the Montagovian framework

With deletion?

Outline

Second-order ACG and Lexical Semantics
Abstract Categorial Grammars
Integrating some lexical semantics information

Parsing ACG
General Idea
Using types

Extended parsers
Typing issues
A new typing system
Example and Datalog

What we would like

Theorem

[Coherence] Let's consider a β-reduced term M and $\langle\Gamma;\gamma\rangle$ its principal typing. If M is almost affine it is the unique β-normal inhabitant of $\langle\Gamma;\gamma\rangle$

Theorem

[Subject Expansion] Let's consider a almost affine term M, a term M' such that $M \twoheadrightarrow_\beta M'$ and $\Gamma \vdash M' : \gamma$. Then $\Gamma \vdash M : \gamma$

What we would like

Theorem

[Subject Expansion] Let's consider a almost affine term M, a term M' such that $M \twoheadrightarrow_\beta M'$ and $\Gamma \vdash M' : \gamma$. Then $\Gamma \vdash M : \gamma$

Typing issues with deletion

Example

- $(\lambda P.\mathbf{c})(\lambda x.\mathbf{fcc}) \twoheadrightarrow_\beta \mathbf{c}$

- $\lambda P.\mathbf{f}((\lambda y.\mathbf{c})(P\mathbf{c})) \twoheadrightarrow_\beta \lambda P.\mathbf{fc}$

Typing issues with deletion

Example

- $(\lambda P.\mathbf{c})(\lambda x.\mathbf{fcc}) \twoheadrightarrow_\beta \mathbf{c}$
  - $\mathbf{c} : a, \mathbf{f} : b \rightarrow b \rightarrow c \vdash (\lambda P.\mathbf{c})(\lambda x.\mathbf{f}xx) : a$
  - $\mathbf{c} : a \vdash \mathbf{c} : a$
- $\lambda P.\mathbf{f}((\lambda y.\mathbf{c})(P\mathbf{c})) \twoheadrightarrow_\beta \lambda P.\mathbf{fc}$
  - $\mathbf{c} : a, \mathbf{f} : a \rightarrow b \vdash \lambda P.\mathbf{f}((\lambda y.\mathbf{c})(P\mathbf{c})) : (a \rightarrow c) \rightarrow b$
  - $\mathbf{c} : a, \mathbf{f} : a \rightarrow b \vdash \lambda P.\mathbf{fc} : o \rightarrow b$

Typing issues with deletion

Example

- $(\lambda P.\mathbf{c})(\lambda x.\mathbf{fcc}) \twoheadrightarrow_\beta \mathbf{c}$
  - $\mathbf{c} : a, \mathbf{f} : b \to b \to c \vdash (\lambda P.\mathbf{c})(\lambda x.\mathbf{f}xx) : a$
  - $\mathbf{c} : a \vdash \mathbf{c} : a$
- $\lambda P.\mathbf{f}((\lambda y.\mathbf{c})(P\mathbf{c})) \twoheadrightarrow_\beta \lambda P.\mathbf{fc}$
  - $\mathbf{c} : a, \mathbf{f} : a \to b \vdash \lambda P.\mathbf{f}((\lambda y.\mathbf{c})(P\mathbf{c})) : (a \to c) \to b$
  - $\mathbf{c} : a, \mathbf{f} : a \to b \vdash \lambda P.\mathbf{fc} : o \to b$

1. Need to include all possible free variables (*i.e.* constants in the case of HOS)
2. Need to know type structure (*skeleton*) for each variable.

Intersection Types

- $(\lambda P.\mathbf{c})(\lambda x.\mathbf{f}xx) \twoheadrightarrow_\beta \mathbf{c}$
  - $\mathbf{c} : a, \mathbf{f} : b \to b \to c \vdash (\lambda P.\mathbf{c})(\lambda x.\mathbf{f}xx) : a$
  - $\mathbf{c} : a \vdash \mathbf{c} : a$
  - We do not know the type of **f**
  - Idea: use intersection types to enumerate possible types in the signature: $\mathbf{f} : (b \to b \to c) \cap (a \to b \to c) \cap \ldots$

Intersection Types

- $(\lambda P.\mathbf{c})(\lambda x.\mathbf{f}xx) \twoheadrightarrow_\beta \mathbf{c}$
  - $\mathbf{c} : a, \mathbf{f} : b \to b \to c \vdash (\lambda P.\mathbf{c})(\lambda x.\mathbf{f}xx) : a$
  - $\mathbf{c} : a \vdash \mathbf{c} : a$
  - We do not know the type of **f**
  - Idea: use intersection types to enumerate possible types in the signature: $\mathbf{f} : (b \to b \to c) \cap (a \to b \to c) \cap \ldots$
- $\lambda P.\mathbf{f}(\lambda y.\mathbf{c}(P\mathbf{c})) \twoheadrightarrow_\beta \lambda P.\mathbf{fc}$

Intersection Types

- $(\lambda P.\mathbf{c})(\lambda x.\mathbf{f}xx) \twoheadrightarrow_\beta \mathbf{c}$
  - $\mathbf{c} : a, \mathbf{f} : b \to b \to c \vdash (\lambda P.\mathbf{c})(\lambda x.\mathbf{f}xx) : a$
  - $\mathbf{c} : a \vdash \mathbf{c} : a$
  - We do not know the type of **f**
  - Idea: use intersection types to enumerate possible types in the signature: $\mathbf{f} : (b \to b \to c) \cap (a \to b \to c) \cap \ldots$
- $\lambda P.\mathbf{f}(\lambda y.\mathbf{c}(P\mathbf{c})) \twoheadrightarrow_\beta \lambda P.\mathbf{fc}$
  - $\mathbf{c} : a, \mathbf{f} : a \to b \vdash \lambda P.\mathbf{f}(\lambda y.\mathbf{c}(P\mathbf{c})) : (a \to c) \to b$
  - $\mathbf{c} : a, \mathbf{f} : a \to b \vdash \lambda P.\mathbf{fc} : o \to b$
  - We do not know the type of P
  - Idea: use intersection types to enumerate possible types in the signature: $P : (a \to c) \cap (a \to b) \cap \ldots$

Intersection Types

Moreover, intersection types are already present (but hidden) in Kanazawa's technique:

$$\exists(\lambda x.\wedge(\mathit{CAKE}\ x)\ (\wedge\ (\mathit{BUY}\ x\ \mathit{MARY})\ (\mathit{EAT}\ x\ \mathit{MARY})))$$

- The two occurrences of *MARY* come from the same lexical entry ($\mathscr{H}_{sem}(\mathit{Mary})$)
- The two occurrences of $\wedge$ come from two different lexical entries ($\mathscr{H}_{sem}(\mathit{and})$ and $\mathscr{H}_{sem}(a)$)
- "Pseudo-principal typing":
  $\mathit{MARY} : a, \wedge : (b_1 \to b_2 \to c_2) \cap (c_1 \to c_2 \to d), \ldots$

Outline

Second-order ACG and Lexical Semantics
- Abstract Categorial Grammars
- Integrating some lexical semantics information

Parsing ACG
- General Idea
- Using types

Extended parsers
- Typing issues
- A new typing system
- Example and Datalog

Restricted intersection types

Rigid variables

A rigid variable x^s is such that x is a variable and s a type skeleton

- Type skeletons: o, $(o \to o) \to o$
- Any type: $s \cdot [\alpha]$
  - $(o \to o) \to o \cdot [a_1, a_2, a_3] = (a_1 \to a_2) \to a_3$

Listed Types

- $\mathscr{T}(\mathscr{A}) ::= \mathscr{A} \mid \mathscr{A} \to \mathscr{T}(\mathscr{A})$
- $\mathscr{T}_s(\mathscr{A})$: simple types of skeletons s
- $\mathscr{L}_s(\mathscr{A}) ::= \mathscr{T}_s(\mathscr{A}) \mid \mathscr{L}_s(\mathscr{A}) \cap \mathscr{L}_s(\mathscr{A})$
- $\mathscr{L}(\mathscr{A}) = \bigcup_s \mathscr{L}_s(\mathscr{A})$
  - Listed types are noted $\overline{\alpha}, \ldots$ and we note $\alpha \in \overline{\alpha}$

Restricted intersection types

Rigid variables

A rigid variable x^s is such that x is a variable and s a type skeleton

- Type skeletons: o, $(o \rightarrow o) \rightarrow o$
- Any type: $s \cdot [\alpha]$
  - $(o \rightarrow o) \rightarrow o \cdot [a_1, a_2, a_3] = (a_1 \rightarrow a_2) \rightarrow a_3$

Listed Types

- $\mathscr{T}(\mathscr{A}) ::= \mathscr{A} \mid \mathscr{A} \rightarrow \mathscr{T}(\mathscr{A})$
- $\mathscr{T}_s(\mathscr{A})$: simple types of skeletons s
- $\mathscr{L}_s(\mathscr{A}) ::= \mathscr{T}_s(\mathscr{A}) \mid \mathscr{L}_s(\mathscr{A}) \cap \mathscr{L}_s(\mathscr{A})$
- $\mathscr{L}(\mathscr{A}) = \bigcup_s \mathscr{L}_s(\mathscr{A})$
  - Listed types are noted $\overline{\alpha}, \ldots$ and we note $\alpha \in \overline{\alpha}$

Restricted intersection types

Rigid variables

A rigid variable x^s is such that x is a variable and s a type skeleton

- Type skeletons: o, $(o \to o) \to o$
- Any type: $s \cdot [\alpha]$
  - $(o \to o) \to o \cdot [a_1, a_2, a_3] = (a_1 \to a_2) \to a_3$

Listed Types

- $\mathscr{T}(\mathscr{A}) ::= \mathscr{A} \mid \mathscr{A} \to \mathscr{T}(\mathscr{A})$
- $\mathscr{T}_s(\mathscr{A})$: simple types of skeletons s
- $\mathscr{L}_s(\mathscr{A}) ::= \mathscr{T}_s(\mathscr{A}) \mid \mathscr{L}_s(\mathscr{A}) \cap \mathscr{L}_s(\mathscr{A})$
- $\mathscr{L}(\mathscr{A}) = \bigcup_s \mathscr{L}_s(\mathscr{A})$
  - Listed types are noted $\overline{\sigma}, \dots$ and we note $\alpha \in \overline{\sigma}$

Restricted intersection types

Rigid variables

A rigid variable x^s is such that x is a variable and s a type skeleton

- Type skeletons: o, $(o \to o) \to o$
- Any type: $s \cdot [\alpha]$
  - $(o \to o) \to o \cdot [a_1, a_2, a_3] = (a_1 \to a_2) \to a_3$

Listed Types

- $\mathscr{T}(\mathscr{A}) ::= \mathscr{A} \mid \mathscr{A} \to \mathscr{T}(\mathscr{A})$
- $\mathscr{T}_s(\mathscr{A})$: simple types of skeletons s
- $\mathscr{L}_s(\mathscr{A}) ::= \mathscr{T}_s(\mathscr{A}) \mid \mathscr{L}_s(\mathscr{A}) \cap \mathscr{L}_s(\mathscr{A})$
- $\mathscr{L}(\mathscr{A}) = \bigcup_s \mathscr{L}_s(\mathscr{A})$
  - Listed types are noted $\overline{\alpha}, \ldots$ and we note $\alpha \in \overline{\alpha}$

Restricted intersection types

Rigid variables

A rigid variable x^s is such that x is a variable and s a type skeleton

- Type skeletons: o, $(o \to o) \to o$
- Any type: $s \cdot [\alpha]$
  - $(o \to o) \to o \cdot [a_1, a_2, a_3] = (a_1 \to a_2) \to a_3$

Listed Types

- $\mathscr{T}(\mathscr{A}) ::= \mathscr{A} \mid \mathscr{A} \to \mathscr{T}(\mathscr{A})$
- $\mathscr{T}_s(\mathscr{A})$: simple types of skeletons s
- $\mathscr{L}_s(\mathscr{A}) ::= \mathscr{T}_s(\mathscr{A}) \mid \mathscr{L}_s(\mathscr{A}) \cap \mathscr{L}_s(\mathscr{A})$
- $\mathscr{L}(\mathscr{A}) = \bigcup_s \mathscr{L}_s(\mathscr{A})$
  - Listed types are noted $\overline{\alpha}, \ldots$ and we note $\alpha \in \overline{\alpha}$

Listed Higher-order Signature

Definition

$\Sigma = (\mathscr{A}, \mathscr{C}, \tau)$

- $\mathscr{A}$ a finite set of atomic types
- $\mathscr{C}$ a finite set of constants
- τ the typing function $\mathscr{C} \to \mathscr{L}(\mathscr{A})$

Listed Higher-order Signature

Definition

$\Sigma = (\mathscr{A}, \mathscr{C}, \tau)$

- $\mathscr{A}$ a finite set of atomic types
- $\mathscr{C}$ a finite set of constants
- τ the typing function $\mathscr{C} \to \mathscr{L}(\mathscr{A})$

Derivations

$$\frac{}{x^s : s \cdot [\alpha] \vdash_\Sigma x^s : s \cdot [\alpha]} \qquad \frac{\alpha \in \tau(\boldsymbol{c})}{\vdash_\Sigma \boldsymbol{c} : \alpha}$$

Listed Higher-order Signature

Definition

$\Sigma = (\mathscr{A}, \mathscr{C}, \tau)$

- $\mathscr{A}$ a finite set of atomic types
- $\mathscr{C}$ a finite set of constants
- τ the typing function $\mathscr{C} \to \mathscr{L}(\mathscr{A})$

Derivations

$$\frac{}{x^s : s \cdot [\alpha] \vdash_\Sigma x^s : s \cdot [\alpha]} \qquad \frac{\alpha \in \tau(\boldsymbol{c})}{\vdash_\Sigma \boldsymbol{c} : \alpha}$$

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

Listed Higher-order Signature

Definition

$\Sigma = (\mathscr{A}, \mathscr{C}, \tau)$

- $\mathscr{A}$ a finite set of atomic types
- $\mathscr{C}$ a finite set of constants
- τ the typing function $\mathscr{C} \to \mathscr{L}(\mathscr{A})$

Derivations

$$\frac{}{x^s : s \cdot [\alpha] \vdash_\Sigma x^s : s \cdot [\alpha]} \qquad \frac{\alpha \in \tau(\boldsymbol{c})}{\vdash_\Sigma \boldsymbol{c} : \alpha}$$

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

$$\frac{\vdash M : \overline{\alpha_1} \quad \vdash M : \overline{\alpha_2}}{\vdash M : \overline{\alpha_1} \cap \overline{\alpha_2}}$$

Characteristic typing

The most general signature for M

- Given $M \in \Lambda_\Sigma$ where $\Sigma = (\mathscr{A}, \mathscr{C}, \tau)$ and $\vdash_\Sigma M : \alpha$ **principal simple type**

  $$\Sigma_M = (\mathscr{A} \cup \{\omega\}, \mathscr{C}, \tau_M) \text{ such that:}$$

  - if $\mathbf{c} \in \mathscr{C}$ in $M \Rightarrow \tau_M(\mathbf{c}) = \tau(\mathbf{c})$
  - otherwise, for $\tau(\mathbf{c}) \in \mathscr{L}_s(\mathscr{A})$,
    $\tau_M(\mathbf{c}) = \bigcap_{(a_1,\ldots,a_{n-1}) \in (\mathscr{A} \cup \{\omega\})^{n-1}} s \cdot [a_1, \ldots, a_{n-1}, \omega]$

Characteristic typing

The most general signature for M

- Given $M \in \Lambda_\Sigma$ where $\Sigma = (\mathscr{A}, \mathscr{C}, \tau)$ and $\vdash_\Sigma M : \alpha$ **principal simple type**

  $\Sigma_M = (\mathscr{A} \cup \{\omega\}, \mathscr{C}, \tau_M)$ such that:
  - if $\mathbf{c} \in \mathscr{C}$ in $M \Rightarrow \tau_M(\mathbf{c}) = \tau(\mathbf{c})$
  - otherwise, for $\tau(\mathbf{c}) \in \mathscr{L}_s(\mathscr{A})$, $\tau_M(\mathbf{c}) = \bigcap_{(a_1,\ldots,a_{n-1}) \in (\mathscr{A} \cup \{\omega\})^{n-1}} s \cdot [a_1, \ldots, a_{n-1}, \omega]$

Characteristic typing

If $\vdash_\Sigma M : \alpha$ is M's principal typing, we can build Σ_M **minimal in** $|\mathscr{A}|$ and obtain $\vdash_{\Sigma_M} M : \overline{\alpha}$, where $\overline{\alpha} = \alpha_1 \cap \ldots \alpha_n$ and n maximal as follows:

Example $\mathscr{C} = \{c_1, c_2, c_3\}$

- Principal on Simple Types:
  - $\tau(c_1) = (a \to u \to b) \to d, \tau(c_2) = a \to a \to b \vdash_\Sigma \lambda x.\boldsymbol{c_1}(\lambda x_1 x_2.\boldsymbol{c_2} x_1 x_1) : u' \to d$

Example $\mathscr{C} = \{c_1, c_2, c_3\}$

- Principal on Simple Types:
  - $\tau(c_1) = (a \rightarrow u \rightarrow b) \rightarrow d, \tau(c_2) = a \rightarrow a \rightarrow b \vdash_\Sigma$ $\lambda x.\boldsymbol{c_1}(\lambda x_1 x_2.\boldsymbol{c_2} x_1 x_1) : u' \rightarrow d$
- Principal with Rigid Variables:
  - $\tau(c_1) = (a \rightarrow (u_3 \rightarrow u_4) \rightarrow b) \rightarrow d, \tau(c_2) = a \rightarrow a \rightarrow b \vdash_\Sigma$ $\lambda x^{o \rightarrow o}.\boldsymbol{c_1}(\lambda x_1^{o} x_2^{o \rightarrow o}.\boldsymbol{c_2} x_1 x_1) : (u_1 \rightarrow u_2) \rightarrow d$

Example $\mathscr{C} = \{c_1, c_2, c_3\}$

- Principal with Rigid Variables:
  - $\tau(c_1) = (a \to (u_3 \to u_4) \to b) \to d, \tau(c_2) = a \to a \to b \vdash_\Sigma \lambda x^{o \to o}.\boldsymbol{c_1}(\lambda x_1^o x_2^{o \to o}.\boldsymbol{c_2} x_1 x_1) : (u_1 \to u_2) \to d$
- Characteristic Typing:
  - $\tau(c_1) = \overline{\alpha_1}, \tau(c_2) = \overline{\alpha_2}, \tau(c_3) = \overline{\alpha_3} \vdash_\Sigma \lambda x^{o \to o}.\boldsymbol{c_1}(\lambda x_1^o x_2^{o \to o}.\boldsymbol{c_2} x_1 x_1) : \overline{\alpha}$
    - $\overline{\alpha_1} = \bigcap_{t \in \mathscr{A}_\omega} (a \to (t \to \omega) \to b) \to d$
    - $\overline{\alpha_2} = a \to a \to b$
    - $\overline{\alpha_3} = \bigcap_{t \in \mathscr{A}_\omega} t \to \omega$
    - $\overline{\alpha} = \bigcap_{t \in \mathscr{A}_\omega} (t \to \omega) \to d$

Example $\mathscr{C} = \{c_1, c_2, c_3\}$

- Characteristic Typing:
  - $\tau(c_1) = \overline{\alpha_1}, \tau(c_2) = \overline{\alpha_2}, \tau(c_3) = \overline{\alpha_3} \vdash_\Sigma \lambda x^{o \to o}.\boldsymbol{c_1}(\lambda x_1^o x_2^{o \to o}.\boldsymbol{c_2} x_1 x_1) : \overline{\alpha}$
    - $\overline{\alpha_1} = \bigcap_{t \in \mathscr{A}_\omega} (a \to (t \to \omega) \to b) \to d$
    - $\overline{\alpha_2} = a \to a \to b$
    - $\overline{\alpha_3} = \bigcap_{t \in \mathscr{A}_\omega} t \to \omega$
    - $\overline{\alpha} = \bigcap_{t \in \mathscr{A}_\omega} (t \to \omega) \to d$

Potentially negatively non-duplicating typing

- $\overline{\alpha_1} = \bigcap_{t \in \mathscr{A}} (a \to (t \to \omega) \to b) \to d$
- $\overline{\alpha_2} = a \to a \to b$
- $\overline{\alpha_3} = \bigcap_{t \in \mathscr{A}} t \to \omega$
- $\overline{\alpha} = \bigcap_{t \in \mathscr{A}} (t \to \omega) \to d$

Potentially negatively non-duplicating typing

Useful occurrences of atomic types

- $\overline{\alpha_1} = \bigcap_{t \in \mathscr{A}} (a^- \to (t \to \omega) \to b^+) \to d^-$
- $\overline{\alpha_2} = a^+ \to a^+ \to b^-$
- $\overline{\alpha_3} = \bigcap_{t \in \mathscr{A}} t \to \omega$
- $\overline{\alpha} = \bigcap_{t \in \mathscr{A}} (t \to \omega) \to d^+$

Such a typing is called a PN-typing

Potentially negatively non-duplicating typing

Useful occurrences of atomic types

- $\overline{\alpha_1} = \bigcap_{t \in \mathscr{A}} (a^- \rightarrow (t \rightarrow \omega) \rightarrow b^+) \rightarrow d^-$
- $\overline{\alpha_2} = a^+ \rightarrow a^+ \rightarrow b^-$
- $\overline{\alpha_3} = \bigcap_{t \in \mathscr{A}} t \rightarrow \omega$
- $\overline{\alpha} = \bigcap_{t \in \mathscr{A}} (t \rightarrow \omega) \rightarrow d^+$

Such a typing is called a PN-typing

Theorem

If a term M is in long-normal form for a PN-typing $\langle \overline{\Gamma}; \overline{\gamma} \rangle$ it is the unique long-normal inhabitant of this pair.

Potentially negatively non-duplicating typing

Useful occurrences of atomic types

- $\overline{\alpha_1} = \bigcap_{t\in\mathscr{A}}(a^- \to (t \to \omega) \to b^+) \to d^-$
- $\overline{\alpha_2} = a^+ \to a^+ \to b^-$
- $\overline{\alpha_3} = \bigcap_{t\in\mathscr{A}} t \to \omega$
- $\overline{\alpha} = \bigcap_{t\in\mathscr{A}}(t \to \omega) \to d^+$

Such a typing is called a PN-typing

Theorem

If a term M is in long-normal form for a PN-typing $\langle\overline{\Gamma};\overline{\gamma}\rangle$ it is the unique long-normal inhabitant of this pair.

Theorem

An almost affine term has a PN characteristic typing.

Properties

The characteristic typing is the simplest typing of $\vdash_{\Sigma_M} M : \overline{\alpha}$ which ensures:

1. M is the unique inhabitant of it.
2. If an almost affine term $M' \twoheadrightarrow_\beta M$, then $\vdash_{\Sigma_M} M : \overline{\alpha}$

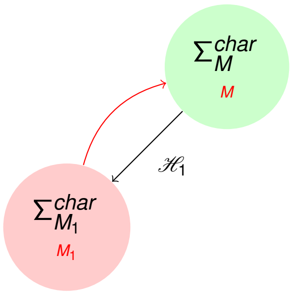
Properties

The characteristic typing is the simplest typing of $\vdash_{\Sigma_M} M : \overline{\alpha}$ which ensures:

1. M is the unique inhabitant of it.
2. If an almost affine term $M' \twoheadrightarrow_\beta M$, then $\vdash_{\Sigma_M} M : \overline{\alpha}$

Moreover, we show almost affine terms M and M' in Λ_{Σ_M} verify $M =_\beta M'$ iff they share the same characteristic typing.

Idea: Use Types



If M_{Σ_1} and $\mathscr{H}_1(M_\Sigma)$ share the same characteristic typing then
$M_{\Sigma_1} =_\beta \mathscr{H}_1(M_\Sigma)$

Outline

Second-order ACG and Lexical Semantics
- Abstract Categorial Grammars
- Integrating some lexical semantics information

Parsing ACG
- General Idea
- Using types

Extended parsers
- Typing issues
- A new typing system
- Example and Datalog

Example

$$READ\ JOHN_{char}\ HAM_{info-cont}$$

IDB

$$\mathcal{L}(John) = \lambda QP.P(Q\ \boldsymbol{JOHN_{char}\ undefined\ undefined})$$
$$\mathcal{L}(Hamlet) = \lambda QP.P(Q\ \boldsymbol{HAM_{char}\ HAM_{phys-obj}\ HAM_{info-cont}})$$
$$\mathcal{L}(read) = \lambda QP.P\pi_1(\lambda x.Q\pi_3\ (\lambda y.\boldsymbol{READ}\ x\ y))$$

$$\pi_i \equiv \lambda x_1 x_2 x_3.x_i$$

Example

EDB

$\boldsymbol{READ}(1,2,3).$
$\boldsymbol{JOHN_{char}}(1).$
$\boldsymbol{HAM_{phys-cont}}(2).$

IDB

$S(x_6)$:- $NP(x_1, x_2, x_3, x_1, x_4, x_5, x_6), NP(y_1, y_2, y_3, y_3, y_4, y_5, x_5), \boldsymbol{READ}(x_4, y_4, y_5).$
$NP(x_1, x_2, x_3, x_4, x_4, x_5, x_5)$:- $\boldsymbol{JOHN_{char}}(x_1), \boldsymbol{undefined}(x_2), \boldsymbol{undefined}(x_3).$
$NP(x_1, x_2, x_3, x_4, x_4, x_5, x_5)$:- $\boldsymbol{HAM_{char}}(x_1), \boldsymbol{HAM_{phys-obj}}(x_2), \boldsymbol{HAM_{info-cont}}(x_3).$

EDB_ω

$\boldsymbol{JOHN_{char}}(\omega).$
$\boldsymbol{HAM_{phys-cont}}(\omega).$
$\boldsymbol{undefined}(\omega).$
$\boldsymbol{MARY_{char}}(\omega).$
$\mathtt{type}(1).$
$\mathtt{type}(2).$
$\mathtt{type}(3).$
$\mathtt{type}(\omega).$

IDB_ω

$\boldsymbol{READ}(x_1, x_2, \omega)$:- $\mathtt{type}(x_1), \mathtt{type}(x_2).$
$\boldsymbol{EAT}(x_1, x_2, \omega)$:- $\mathtt{type}(x_1), \mathtt{type}(x_2).$

$$? :- S(3)$$

Example

IDB

$S(x_6)$:- $NP(x_1, x_2, x_3, x_1, x_4, x_5, x_6)$, $NP(y_1, y_2, y_3, y_3, y_4, y_5, x_5)$, $\boldsymbol{READ}(x_4, y_4, y_5)$.
$NP(x_1, x_2, x_3, x_4, x_4, x_5, x_5)$:- $\boldsymbol{JOHN_{char}}(x_1)$, $\boldsymbol{undefined}(x_2)$, $\boldsymbol{undefined}(x_3)$.
$NP(x_1, x_2, x_3, x_4, x_4, x_5, x_5)$:- $\boldsymbol{HAM_{char}}(x_1)$, $\boldsymbol{HAM_{phys-obj}}(x_2)$, $\boldsymbol{HAM_{info-cont}}(x_3)$.

EDB

$\boldsymbol{READ}(1, 2, 3)$.
$\boldsymbol{JOHN_{char}}(1)$.
$\boldsymbol{HAM_{phys-cont}}(2)$.

IDB_ω

$\boldsymbol{READ}(x_1, x_2, \omega)$:- $\texttt{type}(x_1), \texttt{type}(x_2)$.
$\boldsymbol{EAT}(x_1, x_2, \omega)$:- $\texttt{type}(x_1), \texttt{type}(x_2)$.

EDB_ω

$\boldsymbol{JOHN_{char}}(\omega)$.
$\boldsymbol{HAM_{phys-cont}}(\omega)$.
$\boldsymbol{undefined}(\omega)$.
$\boldsymbol{MARY_{char}}(\omega)$.
$\texttt{type}(1)$.
$\texttt{type}(2)$.
$\texttt{type}(3)$.
$\texttt{type}(\omega)$.

$$? :- S(3)$$

Example

EDB

$\mathbf{READ}(1, 2, 3).$
$\mathbf{JOHN_{char}}(1).$
$\mathbf{HAM_{phys-cont}}(2).$

IDB

$S(x_6)$:- $NP(x_1, x_2, x_3, x_1, x_4, x_5, x_6), NP(y_1, y_2, y_3, y_3, y_4, y_5, x_5), \mathbf{READ}(x_4, y_4, y_5).$
$NP(x_1, x_2, x_3, x_4, x_4, x_5, x_5)$:- $\mathbf{JOHN_{char}}(x_1), \mathbf{undefined}(x_2), \mathbf{undefined}(x_3).$
$NP(x_1, x_2, x_3, x_4, x_4, x_5, x_5)$:- $\mathbf{HAM_{char}}(x_1), \mathbf{HAM_{phys-obj}}(x_2), \mathbf{HAM_{info-cont}}(x_3).$

EDB_ω

$\mathbf{JOHN_{char}}(\omega).$
$\mathbf{HAM_{phys-cont}}(\omega).$
$\mathbf{undefined}(\omega).$
$\mathbf{MARY_{char}}(\omega).$
$\texttt{type}(1).$
$\texttt{type}(2).$
$\texttt{type}(3).$
$\texttt{type}(\omega).$

IDB_ω

$\mathbf{READ}(x_1, x_2, \omega)$:- $\texttt{type}(x_1), \texttt{type}(x_2).$
$\mathbf{EAT}(x_1, x_2, \omega)$:- $\texttt{type}(x_1), \texttt{type}(x_2).$

$?$:- $S(3)$

Conclusion

- Kanazawa: Datalog recognizer for (almost)-linear ACG: efficient parsing (LOGCFL)
  - Result extended to almost affine ACG; at least polynomial time
  - A more complex typing system is needed (intersection which are used in [Sal10])
  - Principal Typings replaced with Charateristic Typing.
- Deletion can be used to enrich the grammar with:
  - Aspects (lexical semantics)
  - Agreement (syntax)
  - ...

Future work

- Parsing:
  - Check magic-set rewriting to lead to prefix-correct Earley algorithm [Kan08]
  - Extract derivations: recognizer → parser.
  - Development.
  - From listed HOS to intersected HOS?
- Linguistic Model:
  - Basic treatment.
  - Unable to reject unfelicitous sentences (*"John fished and ate a fast salmon."* (?))

Future work

- Parsing:
  - Check magic-set rewriting to lead to prefix-correct Earley algorithm [Kan08]
  - Extract derivations: recognizer $\rightarrow$ parser.
  - Development.
  - From listed HOS to intersected HOS?
- Linguistic Model:
  - Basic treatment.
  - Unable to reject unfelicitous sentences (*"John fished and ate a fast salmon."* (?))

Future work

- Parsing:
  - Check magic-set rewriting to lead to prefix-correct Earley algorithm [Kan08]
  - Extract derivations: recognizer → parser.
  - Development.
  - From listed HOS to intersected HOS?
- Linguistic Model:
  - Basic treatment.
  - Unable to reject unfelicitous sentences (*"John fished and ate a fast salmon."* (?))

Future work

- Parsing:
  - Check magic-set rewriting to lead to prefix-correct Earley algorithm [Kan08]
  - Extract derivations: recognizer $\rightarrow$ parser.
  - Development.
  - From listed HOS to intersected HOS?
- Linguistic Model:
  - Basic treatment.
  - Unable to reject unfelicitous sentences (*"John fished and ate a fast salmon."* (?))

Future work

- Parsing:
  - Check magic-set rewriting to lead to prefix-correct Earley algorithm [Kan08]
  - Extract derivations: recognizer $\rightarrow$ parser.
  - Development.
  - From listed HOS to intersected HOS?
- Linguistic Model:
  - Basic treatment.
  - Unable to reject unfelicitous sentences (*"John fished and ate a fast salmon."* (?))

Future work

- Parsing:
  - Check magic-set rewriting to lead to prefix-correct Earley algorithm [Kan08]
  - Extract derivations: recognizer → parser.
  - Development.
  - From listed HOS to intersected HOS?
- Linguistic Model:
  - Basic treatment.
  - Unable to reject unfelicitous sentences (*"John fished and ate a fast salmon."* (?))

Future work

- Parsing:
  - Check magic-set rewriting to lead to prefix-correct Earley algorithm [Kan08]
  - Extract derivations: recognizer $\rightarrow$ parser.
  - Development.
  - From listed HOS to intersected HOS?
- Linguistic Model:
  - Basic treatment.
  - Unable to reject unfelicitous sentences (*"John fished and ate a fast salmon."* (?))

Future work

- Parsing:
  - Check magic-set rewriting to lead to prefix-correct Earley algorithm [Kan08]
  - Extract derivations: recognizer $\rightarrow$ parser.
  - Development.
  - From listed HOS to intersected HOS?
- Linguistic Model:
  - Basic treatment.
  - Unable to reject unfelicitous sentences (*"John fished and ate a fast salmon."* (?))

For Further Reading I

T. Aoto.
Uniqueness of normal proofs in implicational intuitionistic logic.
Journal of Logic, Language and Information, 8:217–242, 1999.

A. Babaev and S. Soloviev.
A coherence theorem for canonical morphism in cartesian closed categories.
Journal of Soviet Mathematics, 20:2263 – 2279, 1982.

P. de Groote.
Towards abstract categorial grammars.
In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, pages 148–155, 2001.

M. Kanazawa.
Parsing and generation as Datalog queries.
In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 176–183, Prague, 2007. Association for Computational Linguistics.

For Further Reading II

M. Kanazawa.
A prefix-correct earley recognizer form multiple context-free grammars.
In *TAG+9, Proceedings of the ninth International Workshop on Tree Adjoining Grammars and Related Frameworks*, Tubingen, Germany, June 2008.

J. Lambek.
The mathematics of sentence structure.
Amer. Math. Mon., 65:154–170, 1958.

R. Montague.
The proper treatment of quantification in ordinary english.
Approaches to Natural Language, pages 221–242, 1973.

R. Muskens.
Lambda Grammars and the Syntax-Semantics Interface.
In R. van Rooy and M. Stokhof, editors, *Proceedings of the Thirteenth Amsterdam Colloquium*, pages 150–155, Amsterdam, 2001.

Sylvain Salvati.
On the membership problem for non-linear abstract categorial grammars.
Journal of Logic, Language and Information, 19(2):163–183, 2010.