From Tree Adjoining Grammars to Higher Order Representations of Abstract Meaning Representations via Abstract Categorial Grammars

Rasmus Blanck, Aleksandre Maskharashvili
Centre for Linguistic Theory and Studies in Probability,
University of Göteborg

29 August 2018
Symposium on Logic and Algorithms in Computational Linguistics
Stockholm, Sweden

Motivation

- Abstract Meaning Representation (AMR) (Banarescu et al., 2013)

Motivation

- Abstract Meaning Representation (AMR) (Banarescu et al., 2013)
  - semantic treebank
  - de-languagized (still biased towards English)
  - used for semantic parsing (Artzi, Lee, and Zettlemoyer, 2015) and generation (Flanigan et al., 2016)
  - limitations: (universal) quantification, negation

Motivation

- Abstract Meaning Representation (AMR) (Banarescu et al., 2013)
  - semantic treebank
  - de-languagized (still biased towards English)
  - used for semantic parsing (Artzi, Lee, and Zettlemoyer, 2015) and generation (Flanigan et al., 2016)
  - limitations: (universal) quantification, negation
  - recent developments:
    - AMRs were transformed as FOL formulas (Bos, 2016)
    - AMRs were transformed as HOL formulas modeling event semantics (Stabler, 2018)
    - problems of quantification, negation were overcome ...

Motivation

- Abstract Meaning Representation (AMR) (Banarescu et al., 2013)
  - semantic treebank
  - de-languagized (still biased towards English)
  - used for semantic parsing (Artzi, Lee, and Zettlemoyer, 2015) and generation (Flanigan et al., 2016)
  - limitations: (universal) quantification, negation
  - recent developments:
    - AMRs were transformed as FOL formulas (Bos, 2016)
    - AMRs were transformed as HOL formulas modeling event semantics (Stabler, 2018)
    - problems of quantification, negation were overcome . . .
- Tree Adjoining Grammars (TAGs) (Joshi, Levy, and Takahashi, 1975)

Motivation

- Abstract Meaning Representation (AMR) (Banarescu et al., 2013)
  - semantic treebank
  - de-languagized (still biased towards English)
  - used for semantic parsing (Artzi, Lee, and Zettlemoyer, 2015) and generation (Flanigan et al., 2016)
  - limitations: (universal) quantification, negation
  - recent developments:
    - AMRs were transformed as FOL formulas (Bos, 2016)
    - AMRs were transformed as HOL formulas modeling event semantics (Stabler, 2018)
    - problems of quantification, negation were overcome ...
- Tree Adjoining Grammars (TAGs) (Joshi, Levy, and Takahashi, 1975)
  - more expressive than context-free grammars (CFGs)
  - (arguably) capable of modeling syntax of natural languages
  - polynomial parsing algorithms (like CFGs)
  - used for generation

Motivation

- Abstract Meaning Representation (AMR) (Banarescu et al., 2013)
  - semantic treebank
  - de-languagized (still biased towards English)
  - used for semantic parsing (Artzi, Lee, and Zettlemoyer, 2015) and generation (Flanigan et al., 2016)
  - limitations: (universal) quantification, negation
  - recent developments:
    - AMRs were transformed as FOL formulas (Bos, 2016)
    - AMRs were transformed as HOL formulas modeling event semantics (Stabler, 2018)
    - problems of quantification, negation were overcome ...
- Tree Adjoining Grammars (TAGs) (Joshi, Levy, and Takahashi, 1975)
  - more expressive than context-free grammars (CFGs)
  - (arguably) capable of modeling syntax of natural languages
  - polynomial parsing algorithms (like CFGs)
  - used for generation
- Abstract Categorial Grammars (ACGs) (De Groote, 2001)
  - type-logical grammatical framework
  - encodes grammatical formalisms, including TAG
  - ACG encoding of TAG enjoys polynomial parsing and generation algorithms
  - embodies Curry's tecto/pheno level distinctions
  - inspired by Montague's translation from syntax to semantics (HOL formulas)

Motivation

- Abstract Meaning Representation (AMR) (Banarescu et al., 2013)
  - semantic treebank
  - de-languagized (still biased towards English)
  - used for semantic parsing (Artzi, Lee, and Zettlemoyer, 2015) and generation (Flanigan et al., 2016)
  - limitations: (universal) quantification, negation
  - recent developments:
    - AMRs were transformed as FOL formulas (Bos, 2016)
    - **AMRs were transformed as HOL formulas modeling event semantics** (Stabler, 2018)
    - **problems of quantification, negation were overcome** ...
- Tree Adjoining Grammars (TAGs) (Joshi, Levy, and Takahashi, 1975)
  - more expressive than context-free grammars (CFGs)
  - (arguably) capable of modeling syntax of natural languages
  - polynomial parsing algorithms (like CFGs)
  - used for generation
- Abstract Categorial Grammars (ACGs) (De Groote, 2001)
  - type-logical grammatical framework
  - encodes grammatical formalisms, including TAG
  - **ACG encoding of TAG enjoys polynomial parsing and generation algorithms**
  - embodies Curry's tecto/pheno level distinctions
  - **inspired by Montague's translation from syntax to semantics (HOL formulas)**

AMR

- Based on frames
- Uniquely rooted directed acyclic graph (DAG) with labeled edges and nodes
  - graph nodes encode entities and events (neo-Davidsonian)
  - edges represent relations among entities, events, etc.
- Capable of expressing various phenomena (e.g. coreference)

AMR

- Based on frames
- Uniquely rooted directed acyclic graph (DAG) with labeled edges and nodes
  - graph nodes encode entities and events (neo-Davidsonian)
  - edges represent relations among entities, events, etc.
- Capable of expressing various phenomena (e.g. coreference)
- Problem with expressing universal quantification in DAG (maybe Hilbert's ϵ-terms?)

Example

A boy wants to go / All boys want to / The boy wants to go / ...
- all have same AMR semantics:

$$\begin{matrix}(w/want_{01} : arg_0(b/boy) \\ \quad : arg_1(g/go_{01} : arg_0\ b))\end{matrix} \quad \text{– AMR in PENMAN notation}$$

$$\begin{matrix}\exists w \exists g \exists b\, (instance(w, want_{01}) \wedge instance(g, w) \wedge \\ \quad instance(b, boy) \wedge arg_0(w, b) \wedge arg_1(w, g) \wedge arg_0(g, b))\end{matrix} \quad \text{– AMR in FOL notation}$$

AMR

- Based on frames
- Uniquely rooted directed acyclic graph (DAG) with labeled edges and nodes
  - graph nodes encode entities and events (neo-Davidsonian)
  - edges represent relations among entities, events, etc.
- Capable of expressing various phenomena (e.g. coreference)
- Problem with expressing universal quantification in DAG (maybe Hilbert's ϵ-terms?)
- Stabler (2018): AAMR
  - transform AMR DAG into tree
  - use tree transducers to obtain HOL formulas with events

Example

A boy wants to go / All boys want to / The boy wants to go / ...

- all have same AMR semantics:

$$\begin{array}{l} (w/want_{01} : arg_0(b/boy) \\ \qquad\qquad : arg_1(g/go_{01} : arg_0\ b)) \end{array} \text{ – AMR in PENMAN notation}$$

$$\begin{array}{l} \exists w \exists g \exists b\, (instance(w, want_{01}) \land instance(g, w) \land \\ \qquad instance(b, boy) \land arg_0(w, b) \land arg_1(w, g) \land arg_0(g, b)) \end{array} \text{ – AMR in FOL notation}$$

$\mathbf{most}(\mathbf{boy.pl}, \lambda b \exists w (\mathbf{walk}_{01}.\mathbf{pres}(w) \land : arg_0(w, b)))$ – **Stabler's HOL encoding**

AMR

- Based on frames
- Uniquely rooted directed acyclic graph (DAG) with labeled edges and nodes
  - graph nodes encode entities and events (neo-Davidsonian)
  - edges represent relations among entities, events, etc.
- Capable of expressing various phenomena (e.g. coreference)
- Problem with expressing universal quantification in DAG (maybe Hilbert's ϵ-terms?)
- Stabler (2018): AAMR
  - transform AMR DAG into tree
  - use tree transducers to obtain HOL formulas with events
  - drawback: coreference is lost

Example

A boy wants to go / All boys want to / The boy wants to go / ...

- all have same AMR semantics:

$$\begin{array}{l}(w/want_{01} : arg_0(b/boy) \\ \qquad\qquad : arg_1(g/go_{01} : arg_0\ b))\end{array} \text{ – AMR in PENMAN notation}$$

$$\begin{array}{l}\exists w \exists g \exists b\, (instance(w, want_{01}) \wedge instance(g, w) \wedge \\ \qquad instance(b, boy) \wedge arg_0(w, b) \wedge arg_1(w, g) \wedge arg_0(g, b))\end{array} \text{ – AMR in FOL notation}$$

$\mathbf{most}(\mathbf{boy.pl}, \lambda b \exists w (\mathbf{walk}_{01}.\mathbf{pres}(w) \wedge : arg_0(w, b)))$ – **Stabler's HOL encoding**

Tree-Adjoining Grammar (TAG) (Joshi, Levy, and Takahashi, 1975)

Elementary trees –

Operations on trees –

Generated structures –

Tree-Adjoining Grammar (TAG) (Joshi, Levy, and Takahashi, 1975)

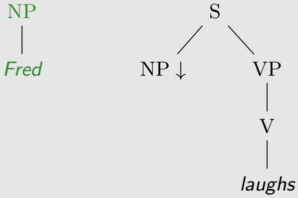
Elementary trees –

- **Initial trees**: domain of locality

Operations on trees –

Generated structures –

Example



Tree-Adjoining Grammar (TAG) (Joshi, Levy, and Takahashi, 1975)

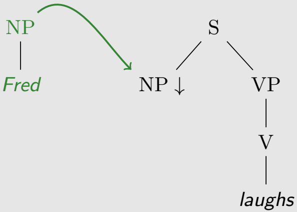
Elementary trees –

- **Initial trees**: domain of locality

Operations on trees – **substitution**

Generated structures –

Example



Tree-Adjoining Grammar (TAG) (Joshi, Levy, and Takahashi, 1975)

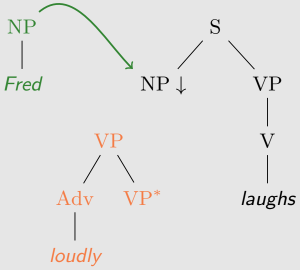
Elementary trees –

- Initial trees: domain of locality
- **Auxiliary trees**: recursion

Operations on trees – substitution

Generated structures –

Example



Tree-Adjoining Grammar (TAG) (Joshi, Levy, and Takahashi, 1975)

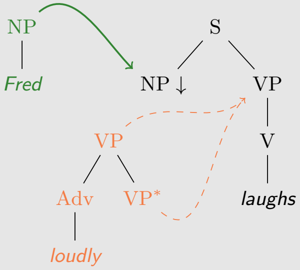
Elementary trees –

- Initial trees: domain of locality
- Auxiliary trees: recursion

Operations on trees – substitution and adjunction

Generated structures –

Example



Tree-Adjoining Grammar (TAG) (Joshi, Levy, and Takahashi, 1975)

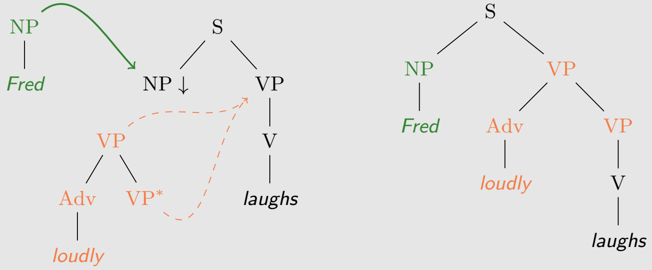
Elementary trees –

- Initial trees: domain of locality
- Auxiliary trees: recursion

Operations on trees – substitution and adjunction

Generated structures – **derived trees.**

Example



Tree-Adjoining Grammar (TAG) (Joshi, Levy, and Takahashi, 1975)

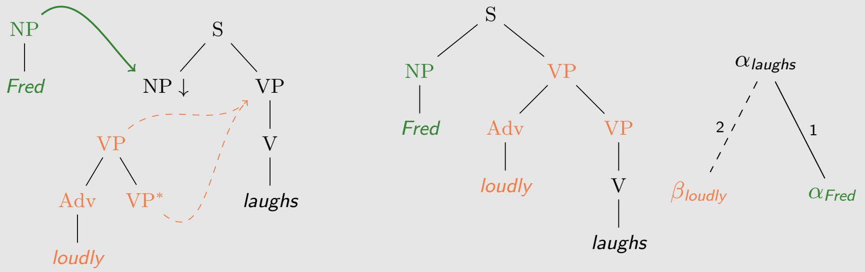
Elementary trees –

- Initial trees: domain of locality
- Auxiliary trees: recursion

Operations on trees – substitution and adjunction

Generated structures – derived trees. Their by-products : **derivation trees**

Example



1 **ACG**

- ACG definition

Abstract Categorial Grammar (ACG)
(De Groote, 2001)

Main Features

- ACGs are a **grammatical framework**

Abstract Categorial Grammar (ACG)
(De Groote, 2001)

Main Features

- ACGs are a **grammatical framework**
- An ACG $\mathscr{G}$ generates **two** languages :
  - The **abstract** language $\mathcal{A}(\mathscr{G})$
  - The **object** language $\mathcal{O}(\mathscr{G})$

Abstract Categorial Grammar (ACG)
(De Groote, 2001)

Main Features

- ACGs are a **grammatical framework**
- An ACG $\mathscr{G}$ generates **two** languages :
  - The **abstract** language $\mathcal{A}(\mathscr{G})$
  - The **object** language $\mathcal{O}(\mathscr{G})$

Abstract language : Admissible structures (parse structures, derivations)
Object language : An **interpretation** of the abstract language

Abstract Categorial Grammar (ACG)
(De Groote, 2001)

Main Features

- ACGs are a **grammatical framework**
- An ACG $\mathscr{G}$ generates **two** languages :
  - The **abstract** language $\mathcal{A}(\mathscr{G})$
  - The **object** language $\mathcal{O}(\mathscr{G})$

Abstract language : Admissible structures (parse structures, derivations)
Object language : An **interpretation** of the abstract language

Basic properties

Modularity Both languages are of the same nature – sets of linear λ-terms

Abstract Categorial Grammar (ACG)
(De Groote, 2001)

Main Features

- ACGs are a **grammatical framework**
- An ACG $\mathscr{G}$ generates **two** languages :
  - The **abstract** language $\mathcal{A}(\mathscr{G})$
  - The **object** language $\mathcal{O}(\mathscr{G})$

Abstract language : Admissible structures (parse structures, derivations)
Object language : An **interpretation** of the abstract language

Basic properties

Modularity Both languages are of the same nature – sets of linear λ-terms :
ACGs can be composed

Abstract Categorial Grammar (ACG)
(De Groote, 2001)

Main Features

- ACGs are a **grammatical framework**
- An ACG $\mathscr{G}$ generates **two** languages :
  - The **abstract** language $\mathcal{A}(\mathscr{G})$
  - The **object** language $\mathcal{O}(\mathscr{G})$

Abstract language : Admissible structures (parse structures, derivations)
Object language : An **interpretation** of the abstract language

Basic properties

Modularity Both languages are of the same nature – sets of linear λ-terms :
ACGs can be composed

Parsing 2nd order ACGs are **reversible** (Salvati, 2005; Kanazawa, 2007)

ACG definition

Definition (ACG)

An abstract categorial grammar (ACG) $\mathscr{G}$ is a quadruple $\langle \Sigma_1, \Sigma_2, \mathscr{L}, s \rangle$, where

1. Σ_1 and Σ_2 are higher-order linear signatures, called the *abstract* vocabulary and the *object* vocabulary, respectively;
2. $\mathscr{L} : \Sigma_1 \longrightarrow \Sigma_2$ is a lexicon; $\mathscr{L}(\lambda x.M) = \lambda x.\mathscr{L}(M)$ and $\mathscr{L}(M\,N) = \mathscr{L}(M)\,\mathscr{L}(N)$
3. s is a type of the abstract vocabulary (either atomic or built upon the atomic types in Σ_1), called the *distinguished* type of the grammar.

ACG definition

Definition (ACG)

An abstract categorial grammar (ACG) $\mathscr{G}$ is a quadruple $\langle \Sigma_1, \Sigma_2, \mathscr{L}, s \rangle$, where

1. Σ_1 and Σ_2 are higher-order linear signatures, called the *abstract* vocabulary and the *object* vocabulary, respectively;
2. $\mathscr{L} : \Sigma_1 \longrightarrow \Sigma_2$ is a lexicon; $\mathscr{L}(\lambda x.M) = \lambda x.\mathscr{L}(M)$ and $\mathscr{L}(M\,N) = \mathscr{L}(M)\,\mathscr{L}(N)$
3. s is a type of the abstract vocabulary (either atomic or built upon the atomic types in Σ_1), called the *distinguished* type of the grammar.

The **abstract language**: $\mathcal{A}(\mathscr{G}) = \{M \in \Lambda(\Sigma_1) \mid \ \vdash_{\Sigma_1} M : s \quad \text{is derivable}\}$

ACG definition

Definition (ACG)

An abstract categorial grammar (ACG) $\mathscr{G}$ is a quadruple $\langle \Sigma_1, \Sigma_2, \mathscr{L}, s \rangle$, where

1. Σ_1 and Σ_2 are higher-order linear signatures, called the *abstract* vocabulary and the *object* vocabulary, respectively;
2. $\mathscr{L} : \Sigma_1 \longrightarrow \Sigma_2$ is a lexicon; $\mathscr{L}(\lambda x.M) = \lambda x.\mathscr{L}(M)$ and $\mathscr{L}(M\,N) = \mathscr{L}(M)\,\mathscr{L}(N)$
3. s is a type of the abstract vocabulary (either atomic or built upon the atomic types in Σ_1), called the *distinguished* type of the grammar.

The **abstract language**: $\mathcal{A}(\mathscr{G}) = \{M \in \Lambda(\Sigma_1) \mid \ \vdash_{\Sigma_1} M : s \quad \text{is derivable}\}$
The **object language**: $\mathcal{O}(\mathscr{G}) = \{N \in \Lambda(\Sigma_2) \mid \exists M \in \mathscr{A}(\mathscr{G}) : N = \mathscr{L}(M)\}$

ACG definition

Definition (ACG)

An abstract categorial grammar (ACG) $\mathscr{G}$ is a quadruple $\langle \Sigma_1, \Sigma_2, \mathscr{L}, s \rangle$, where

1. Σ_1 and Σ_2 are higher-order linear signatures, called the *abstract* vocabulary and the *object* vocabulary, respectively;
2. $\mathscr{L} : \Sigma_1 \longrightarrow \Sigma_2$ is a lexicon; $\mathscr{L}(\lambda x.M) = \lambda x.\mathscr{L}(M)$ and $\mathscr{L}(M\,N) = \mathscr{L}(M)\,\mathscr{L}(N)$
3. s is a type of the abstract vocabulary (either atomic or built upon the atomic types in Σ_1), called the *distinguished* type of the grammar.

The **abstract language**: $\mathcal{A}(\mathscr{G}) = \{M \in \Lambda(\Sigma_1) \mid \vdash_{\Sigma_1} M : s \quad \text{is derivable}\}$
The **object language**: $\mathcal{O}(\mathscr{G}) = \{N \in \Lambda(\Sigma_2) \mid \exists M \in \mathscr{A}(\mathscr{G}) : N = \mathscr{L}(M)\}$

Modularity: ACGs can be composed as lexicons are functions.

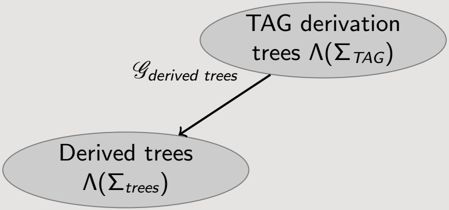
TAG as ACGs

TAG derivation trees $\Lambda(\Sigma_{TAG})$

TAG as ACGs

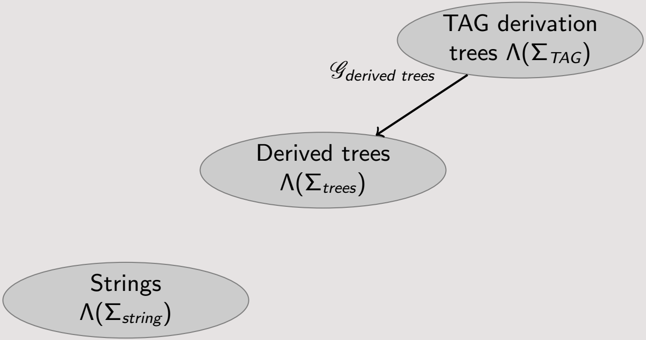

TAG derivation trees $\Lambda(\Sigma_{TAG})$
Derived trees $\Lambda(\Sigma_{trees})$

TAG as ACGs

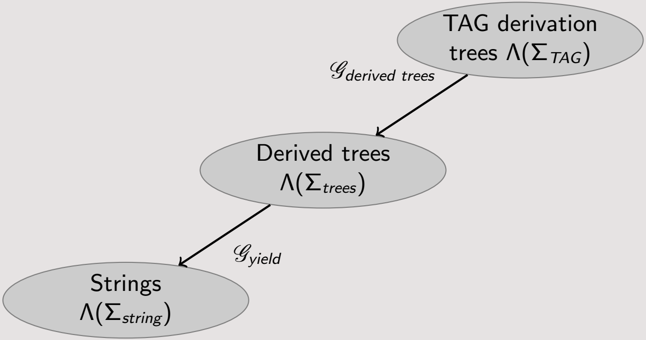


TAG derivation trees $\Lambda(\Sigma_{TAG})$
$\mathscr{G}_{derived\ trees}$
Derived trees $\Lambda(\Sigma_{trees})$

TAG as ACGs

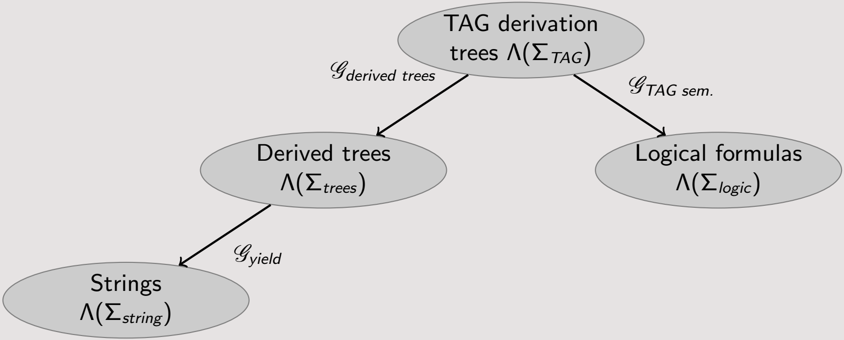

TAG derivation trees $\Lambda(\Sigma_{TAG})$
$\mathscr{G}_{derived\ trees}$
Derived trees $\Lambda(\Sigma_{trees})$
Strings $\Lambda(\Sigma_{string})$

TAG as ACGs



TAG derivation trees $\Lambda(\Sigma_{TAG})$
$\mathscr{G}_{derived\ trees}$
Derived trees $\Lambda(\Sigma_{trees})$
$\mathscr{G}_{yield}$
Strings $\Lambda(\Sigma_{string})$

TAG as ACGs + Montague semantics (Pogodalla, 2004a)



From TAG derivation to TAG derived trees

Derivation trees

Their interpretations as derived trees

From TAG derivation to TAG derived trees

Derivation trees

Their interpretations as derived trees

$$\begin{array}{c}\mathrm{NP}\\ |\\ \textit{Fred}\end{array}$$

From TAG derivation to TAG derived trees

Derivation trees

C_{Fred} : **NP**

Their interpretations as derived trees

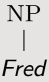
NP_1 *Fred*

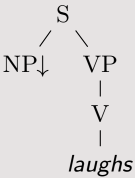
$$\begin{array}{c}\mathrm{NP}\\ |\\ \textit{Fred}\end{array}$$

From TAG derivation to TAG derived trees

Derivation trees
C_{Fred} : $\mathbf{NP}$
C_{laughs} : $\mathbf{S}_{\mathsf{A}} \multimap \mathbf{VP}_{\mathsf{A}} \multimap \mathbf{NP} \multimap \mathbf{S}$

Their interpretations as derived trees
NP_1 *Fred*
$\lambda\, \mathbf{a}^{\mathbf{S}}\, \mathbf{a}^{\mathbf{V}}\, \mathbf{np}.\, \mathbf{a}^{\mathbf{S}}\, (\mathrm{S}_2\ \mathbf{np}\ (\mathbf{a}^{\mathbf{V}}\ (\mathrm{VP}_2\ (\mathrm{V}_1\ \textit{laughs}))))$





From TAG derivation to TAG derived trees

Derivation trees

$C_{\mathit{Fred}} : \mathbf{NP}$

$C_{\mathit{laughs}} : \mathbf{S}_{\mathsf{A}} \multimap \mathbf{VP}_{\mathsf{A}} \multimap \mathbf{NP} \multimap \mathbf{S}$

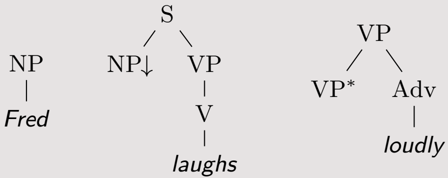
$C_{\mathit{loudly}} : \mathbf{VP}_{\mathsf{A}} \multimap \mathbf{VP}_{\mathsf{A}}$

Their interpretations as derived trees

NP_1 *Fred*

$\lambda\, \mathbf{a}^{\mathbf{S}}\, \mathbf{a}^{\mathbf{V}}\, \mathbf{np}.\, \mathbf{a}^{\mathbf{S}}\, (\mathrm{S}_2\ \mathbf{np}\ (\mathbf{a}^{\mathbf{V}}\ (\mathrm{VP}_2\ (\mathrm{V}_1\ \mathit{laughs}))))$

$\lambda \mathbf{a}^{\mathbf{V}}\, x.\, \mathbf{a}^{\mathbf{V}}\ (\mathrm{V}_2\ x\ (\mathrm{Adv}_1\ \mathit{loudly}))$



From TAG derivation to TAG derived trees

Derivation trees

C_{Fred} : $\mathbf{NP}$

C_{laughs} : $\mathbf{S}_\mathsf{A} \multimap \mathbf{VP}_\mathsf{A} \multimap \mathbf{NP} \multimap \mathbf{S}$

C_{loudly} : $\mathbf{VP}_\mathsf{A} \multimap \mathbf{VP}_\mathsf{A}$

$\mathsf{I}_{\mathbf{X}_\mathsf{A}}$: $\mathbf{X}_\mathsf{A}$

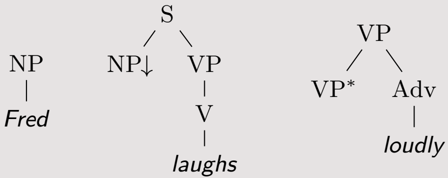
Their interpretations as derived trees

NP_1 *Fred*

$\lambda\, \mathbf{a}^{\mathbf{S}}\, \mathbf{a}^{\mathbf{V}}\, \mathbf{np}.\, \mathbf{a}^{\mathbf{S}}\, (\mathrm{S}_2\ \mathbf{np}\ (\mathbf{a}^{\mathbf{V}}\ (\mathrm{VP}_2\ (\mathrm{V}_1\ \textit{laughs}))))$

$\lambda \mathbf{a}^{\mathbf{V}}\ x.\, \mathbf{a}^{\mathbf{V}}\ (\mathrm{V}_2\ x\ (\mathrm{Adv}_1\ \textit{loudly}))$

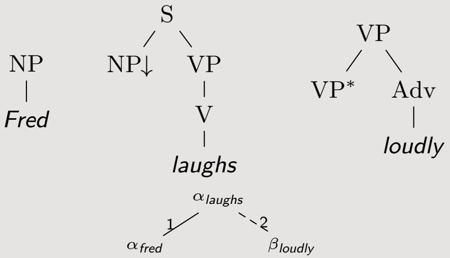
$\lambda x.x$



From TAG derivation to TAG derived trees

Derivation trees
C_{Fred} : $\mathbf{NP}$
C_{laughs} : $\mathbf{S}_\mathsf{A} \multimap \mathbf{VP}_\mathsf{A} \multimap \mathbf{NP} \multimap \mathbf{S}$
C_{loudly} : $\mathbf{VP}_\mathsf{A} \multimap \mathbf{VP}_\mathsf{A}$
$\mathsf{I}_{\mathbf{X}_\mathsf{A}}$: $\mathbf{X}_\mathsf{A}$

Their interpretations as derived trees
NP_1 *Fred*
$\lambda\, \mathbf{a}^\mathsf{S}\, \mathbf{a}^\mathsf{V}\, \mathbf{np}.\, \mathbf{a}^\mathsf{S}\, (\mathrm{S}_2\ \mathbf{np}\ (\mathbf{a}^\mathsf{V}\ (\mathrm{VP}_2\ (\mathrm{V}_1\ \textit{laughs}))))$
$\lambda \mathbf{a}^\mathsf{V}\, x.\, \mathbf{a}^\mathsf{V}\, (\mathrm{V}_2\ x\ (\mathrm{Adv}_1\ \textit{loudly}))$
$\lambda x.x$



$$M_0 = C_{left}\ \mathsf{I}_\mathsf{S}\ (C_{loudly}\ \mathsf{I}_\mathsf{V})\ C_{Fred}$$

From TAG derivation to TAG derived trees

Derivation trees

$C_{Fred} : \mathbf{NP}$

$C_{laughs} : \mathbf{S}_{\mathsf{A}} \multimap \mathbf{VP}_{\mathsf{A}} \multimap \mathbf{NP} \multimap \mathbf{S}$

$C_{loudly} : \mathbf{VP}_{\mathsf{A}} \multimap \mathbf{VP}_{\mathsf{A}}$

$\mathsf{I}_{\mathbf{X}_{\mathsf{A}}} : \mathbf{X}_{\mathsf{A}}$

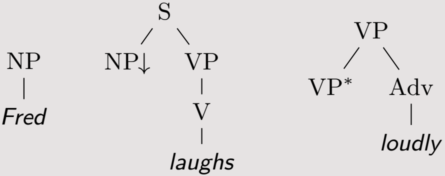
Their interpretations as derived trees

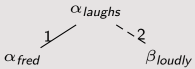
$\mathrm{NP}_1\ \textit{Fred}$

$\lambda\, \mathbf{a^S}\, \mathbf{a^V}\, \mathbf{np}.\, \mathbf{a^S}\, (\mathrm{S}_2\ \mathbf{np}\ (\mathbf{a^V}\ (\mathrm{VP}_2\ (\mathrm{V}_1\ \textit{laughs}))))$

$\lambda \mathbf{a^V}\ x.\, \mathbf{a^V}\ (\mathrm{V}_2\ x\ (\mathrm{Adv}_1\ \textit{loudly}))$

$\lambda x.x$





$$M_0 = C_{left}\ \mathsf{I}_{\mathbf{S}}\ (C_{loudly}\ \mathsf{I}_{\mathbf{V}})\ C_{Fred}$$

$$\mathscr{G}_{yield} \circ \mathscr{G}_{derived\ trees}(M_0) = \textit{Fred} + \textit{loudly} + \textit{laughs}$$

From TAG derivation to Montague Translations (Pogodalla, 2004b)

Derivation trees

Interpretations into Montague Grammar

From TAG derivation to Montague Translations (Pogodalla, 2004b)

Derivation trees

C_{Fred} : **NP**

Interpretations into Montague Grammar

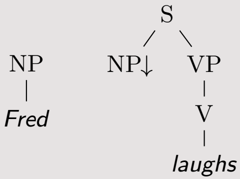
$\lambda P.\, P\, \texttt{fred}$

$$\begin{array}{c}\mathrm{NP}\\ |\\ \textit{Fred}\end{array}$$

From TAG derivation to Montague Translations (Pogodalla, 2004b)

Derivation trees
C_{Fred} : **NP**

Interpretations into Montague Grammar
$\lambda P.\, P\, \texttt{fred}$



From TAG derivation to Montague Translations (Pogodalla, 2004b)

Derivation trees

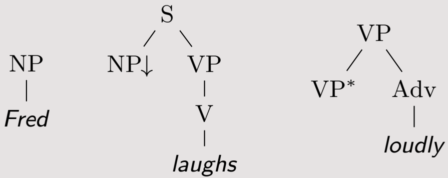
C_{Fred} : $\mathbf{NP}$

C_{laughs} : $\mathbf{S}_{\mathsf{A}} \multimap \mathbf{VP}_{\mathsf{A}} \multimap \mathbf{NP} \multimap \mathbf{S}$

Interpretations into Montague Grammar

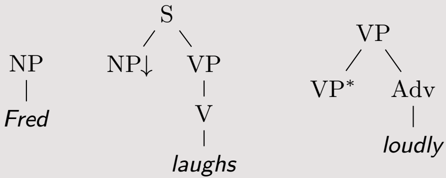
$\lambda P.\, P\, \texttt{fred}$

$\lambda\, \mathbf{a^S}\, \mathbf{a^V}\, \mathbf{np}.\, \mathbf{a^S}\, (\mathbf{np}\, (\mathbf{a^V}\, (\lambda x.\, \texttt{smile}\, x)))$



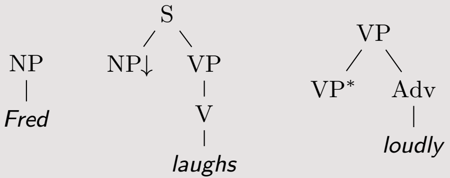
From TAG derivation to Montague Translations (Pogodalla, 2004b)

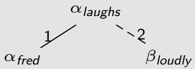
Derivation trees	Interpretations into Montague Grammar
$C_{Fred} : \mathbf{NP}$	$\lambda P.\, P\, \texttt{fred}$
$C_{laughs} : \mathbf{S}_{\mathrm{A}} \multimap \mathbf{VP}_{\mathrm{A}} \multimap \mathbf{NP} \multimap \mathbf{S}$	$\lambda\, \mathbf{a^S}\, \mathbf{a^V}\, \mathbf{np}.\, \mathbf{a^S}\, (\mathbf{np}\, (\mathbf{a^V}\, (\lambda x.\, \texttt{smile}\, x)))$
$C_{loudly} : \mathbf{VP}_{\mathrm{A}} \multimap \mathbf{VP}_{\mathrm{A}}$	$\lambda\, \mathbf{a^V}.\, \mathbf{a^V}\, (\lambda x.\, \texttt{loud}\, x))$



From TAG derivation to Montague Translations (Pogodalla, 2004b)

Derivation trees	Interpretations into Montague Grammar
C_{Fred} : $\mathbf{NP}$	$\lambda P.\, P\ \mathtt{fred}$
C_{laughs} : $\mathbf{S}_{\mathrm{A}} \multimap \mathbf{VP}_{\mathrm{A}} \multimap \mathbf{NP} \multimap \mathbf{S}$	$\lambda\, \mathbf{a}^{\mathbf{S}}\, \mathbf{a}^{\mathbf{V}}\, \mathbf{np}.\, \mathbf{a}^{\mathbf{S}}\, (\mathbf{np}\, (\mathbf{a}^{\mathbf{V}}\, (\lambda x.\, \mathtt{smile}\, x)))$
C_{loudly} : $\mathbf{VP}_{\mathrm{A}} \multimap \mathbf{VP}_{\mathrm{A}}$	$\lambda\, \mathbf{a}^{\mathbf{V}}.\, \mathbf{a}^{\mathbf{V}}\, (\lambda x.\, \mathtt{loud}\, x))$
$\mathrm{I}_{\mathbf{X}_{\mathbf{A}}}$: $\mathbf{X}_{\mathbf{A}}$	$\lambda x.x$

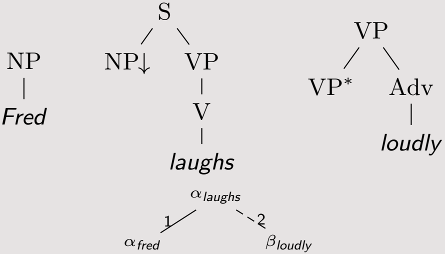




$$M_0 = C_{laughs}\ \mathrm{I}_{\mathbf{S}}\ (C_{loudly}\ \mathrm{I}_{\mathbf{V}})\ C_{Fred}$$

From TAG derivation to Montague Translations (Pogodalla, 2004b)

Derivation trees	Interpretations into Montague Grammar
C_{Fred} : $\mathbf{NP}$	$\lambda P.\, P\, \mathtt{fred}$
C_{laughs} : $\mathbf{S}_{\mathrm{A}} \multimap \mathbf{VP}_{\mathrm{A}} \multimap \mathbf{NP} \multimap \mathbf{S}$	$\lambda\, \mathbf{a^S}\, \mathbf{a^V}\, \mathbf{np}.\, \mathbf{a^S}\, (\mathbf{np}\, (\mathbf{a^V}\, (\lambda x.\, \mathtt{smile}\, x)))$
C_{loudly} : $\mathbf{VP}_{\mathrm{A}} \multimap \mathbf{VP}_{\mathrm{A}}$	$\lambda\, \mathbf{a^V}.\, \mathbf{a^V}\, (\lambda x.\, \mathtt{loud}\, x))$
$\mathrm{I}_{\mathbf{X_A}}$: $\mathbf{X_A}$	$\lambda x.x$



$$M_0 = C_{laughs}\ \mathrm{I}_{\mathbf{S}}\ (C_{loudly}\ \mathrm{I}_{\mathbf{V}})\ C_{Fred}$$

$$\mathscr{L}_{Log}(M_0) = \mathtt{loud}\,(\mathtt{smile\ fred})$$

TAG derivation trees to HOL (Pogodalla, 2017)

Constants of Σ_{TAG}	Their interpretations by $\mathscr{G}_{TAG\ sem.}$
$C_{fred} : \mathbf{NP}$	$\lambda P. P\,\mathtt{fred} : (e \to t) \to t$
$C_{woman} : \mathbf{n_A} \multimap \mathbf{NP}$	$\lambda D.\lambda q\,.D\,\mathtt{woman}\,q$
$C_{smart} : \mathbf{n_A} \multimap \mathbf{n_A}$	$\lambda D.\ \lambda n\,.\lambda q\,.\,D\,(\lambda\,x.\,(\mathtt{smart}\ x) \land (n\ x))q$
$C_{every}, C_{each} : \mathbf{n_A}$	$\lambda\,P\,Q\,.\forall\,x.\,(P\,x)\ \supset\ (Q\,x)$
$C_{some},\ C_a : \mathbf{n_A}$	$\lambda\,P\,Q\,.\exists\,x.\,(P\,x)\ \land\ (Q\,x)$
$C_{kissed} : \mathbf{S}_A \multimap \mathbf{VP}_A \multimap \mathbf{NP} \multimap \mathbf{NP} \multimap \mathbf{S}$	$\lambda adv_s\ adv_v\ sbj\ obj.\ adv_s\,(sbj\,(\lambda x.(obj\,(adv_v(\lambda y.\mathtt{kiss}\,x\,y)))))$
$\mathsf{I}_X : X_A$	$\lambda x.x$
$\mathbf{S}$	t

Continuations, event semantics, ACG

Previous approaches

- syntax-event semantics interface using ACG (Winter and Zwarts, 2011) – their grammar is not TAG;
- syntax-event semantic interface (Champollion, 2015):

Continuations, event semantics, ACG

Previous approaches

- syntax-event semantics interface using ACG (Winter and Zwarts, 2011) – their grammar is not TAG;
- syntax-event semantic interface (Champollion, 2015):
  - uses continuations: verbs are of type $(v \to t) \to t$
  - negation scopes over existentially closed formula $(\neg\exists w \ldots)$
  - no distinction of arguments and adjuncts, e.g.

$$\lambda x.go\ x \text{ VS } \lambda f.\exists w.go(w) \wedge f(w)$$

Continuations, event semantics, ACG

Previous approaches

- syntax-event semantics interface using ACG (Winter and Zwarts, 2011) – their grammar is not TAG;
- syntax-event semantic interface (Champollion, 2015):
  - uses continuations: verbs are of type $(v \to t) \to t$
  - negation scopes over existentially closed formula $(\neg\exists w \ldots)$
  - no distinction of arguments and adjuncts, e.g.

$$\lambda x.go\ x \text{ VS } \lambda f.\exists w.go(w) \wedge f(w)$$

Our approach

- use continuations, like (Champollion, 2015)
- negation scopes over event quantifier, like (Champollion, 2015)
- retain arguments within a lexical entry of a verb, like AMR
- (universal) quantification, like (Stabler, 2018)

Interpretation as HOL formulas modeling event semantics: First try

Interpretation as HOL formulas modeling event semantics: First try

- everything should get a chance for a continuation
- but one has to know when to stop (close)

Interpretation as HOL formulas modeling event semantics: First try

- everything should get a chance for a continuation
- but one has to know when to stop (close)

$$
\begin{array}{ll}
\mathbf{S} & := (v \to t) \to t \\
\mathbf{T} & := t \\
\mathit{Closure} & := \lambda P.P\ \mathit{True}\ :\ ((v \to t) \to t) \to t \\
C_{john} & := \lambda P.\ P\,\mathrm{john} \\
C_{walks} & := \lambda adv_s\ adv_v\ subj.\ adv_s\,(subj\,(adv_v(\lambda x.\lambda h.\exists w.\,(\mathrm{walk}\,w) \wedge (arg_0\,w\,x) \wedge (h\,w))) \\
C_{smart} & := \lambda D.\lambda n.\lambda q.\lambda f.D(\lambda xh.(n\,x\,h) \wedge (\mathrm{smart}\,x))q\,f \\
C^{\mathbf{n_A}}_{every} & := \lambda p.\lambda q.\lambda f.\forall x.(p\,x\,f) \supset (q\,x\,f) \\
C^{\mathbf{n_A} \multimap \mathbf{NP}}_{woman} & := \lambda D.D(\lambda\,x\,h.(\mathrm{woman}\,x \wedge h\,x)) \\
C^{\mathbf{S}_A \multimap \mathbf{S}_A}_{certainly} & := \lambda m.\,\lambda V.\ m\,(\lambda h.V(\lambda v.(\mathrm{certainly}\,v) \wedge (h\,v)) \\
C^{\mathbf{VP}_A \multimap \mathbf{VP}_A}_{fast} & := \lambda m.\,\lambda V.\,m\,(\lambda x.\lambda h.Vx(\lambda v.(\mathrm{fast}\,v) \wedge (h\,v))) \\
C^{\mathbf{VP}_A}_{does\ not} & := \lambda Vxh.\neg(V\,x\,h)
\end{array}
$$

First try: Results

(1) Every smart woman walks.

$M_1 = \mathit{Closure}\,(C_{walks}\,\mathsf{I_S}\,\mathsf{I_{VP}}\,(C_{woman}\,(C_{smart}\;C_{every})))\;\;:\mathbf{T}$

$$M_1 := \quad \forall x(\text{woman}\,x \wedge \text{smart}\,x \supset \exists w\,(\text{walk}\,w) \wedge (\mathit{arg}_0\,w\,x)) \qquad \checkmark$$

First try: Results

(1) Every smart woman walks.
$M_1 = \mathit{Closure}\,(C_{walks}\; \mathsf{I}_{\mathbf{S}}\; \mathsf{I}_{\mathbf{VP}}\; (C_{woman}\; (C_{smart}\; C_{every})))\;\; : \mathbf{T}$

(2) John does not walk.
$M_2 = \mathit{Closure}\,(C_{walks}\; \mathsf{I}_{\mathbf{S}}\;\; C_{does\ not}\;\; C_{john})\;\; : \mathbf{T}$

$M_1 := \quad \forall x(\text{woman}\, x \wedge \text{smart}\, x \supset \exists w\, (\text{walk}\, w) \wedge (arg_0\; w\; x))$ ✓

$M_2 := \quad \neg\exists w\, (\text{walk}\, w) \wedge (arg_0\; w\; \text{john})$ ✓

First try: Results

(1) Every smart woman walks.
$M_1 = \mathit{Closure}\,(C_{walks}\ \mathsf{I_S}\ \mathsf{I_{VP}}\ (C_{woman}\ (C_{smart}\ C_{every})))\ \ : \mathbf{T}$

(2) John does not walk.
$M_2 = \mathit{Closure}\,(C_{walks}\ \mathsf{I_S}\ \ C_{does\ not}\ \ C_{john})\ \ : \mathbf{T}$

(3) Every smart woman walks fast.
$M_3 = \mathit{Closure}\,(C_{walks}\mathsf{I_S}(C_{fast}\ \mathsf{I_{VP}})(C_{woman}\ (C_{smart}\ C_{every})))\ \ : \mathbf{T}$

$M_1 := \quad \forall x(\text{woman}\, x \wedge \text{smart}\, x \supset \exists w\, (\text{walk}\, w) \wedge (\mathit{arg}_0\, w\, x))$ ✓
$M_2 := \quad \neg\exists w\, (\text{walk}\, w) \wedge (\mathit{arg}_0\, w\, \text{john})$ ✓
$M_3 := \quad \forall x(\text{woman}\, x \wedge \text{smart}\, x \wedge \text{fast}\, x \supset \exists w(\text{walk}\, w) \wedge (\mathit{arg}_0\, w\, x) \wedge (\text{fast}\, w))$

First try: Results

(1) Every smart woman walks.
$M_1 = \mathit{Closure}\,(C_{walks}\ \mathsf{I}_{\mathbf{S}}\ \mathsf{I}_{\mathbf{VP}}\ (C_{woman}\ (C_{smart}\ C_{every})))\ \ : \mathbf{T}$

(2) John does not walk.
$M_2 = \mathit{Closure}\,(C_{walks}\ \mathsf{I}_{\mathbf{S}}\ \ C_{does\ not}\ \ C_{john})\ \ : \mathbf{T}$

(3) Every smart woman walks fast.
$M_3 = \mathit{Closure}\,(C_{walks}\mathsf{I}_{\mathbf{S}}(C_{fast}\ \mathsf{I}_{\mathbf{VP}})(C_{woman}\ (C_{smart}\ C_{every})))\ \ : \mathbf{T}$

(4) Certainly, every smart woman walks.
$M_4 = \mathit{Closure}\,(C_{walks}(C_{certainly}\ \mathsf{I}_{\mathbf{S}})\mathsf{I}_{\mathbf{VP}}(C_{woman}\ (C_{smart}\ C_{every})))\ \ : \mathbf{T}$

$M_1 := \quad \forall x(\text{woman}\,x \wedge \text{smart}\,x \supset \exists w\,(\text{walk}\,w) \wedge (\mathit{arg}_0\ w\,x)) \qquad \checkmark$

$M_2 := \quad \neg\exists w\,(\text{walk}\,w) \wedge (\mathit{arg}_0\ w\,\text{john}) \qquad \checkmark$

$M_3 := \quad \forall x(\text{woman}\,x \wedge \text{smart}\,x \wedge \text{fast}\,x \supset \exists w(\text{walk}\,w) \wedge (\mathit{arg}_0\ w\,x) \wedge (\text{fast}\,w))$

$M_4 := \quad \forall x\,(\text{woman}\,x \wedge \text{smart}\,x \wedge \text{certainly}\,x \supset \exists w(\text{walk}\,w) \wedge (\mathit{arg}_0\ w\,x) \wedge (\text{certainly}\,w))$

Locating the problem

$$
\begin{array}{ll}
\mathbf{S} & := (v \to t) \to t \\
\mathbf{T} & := t \\
\mathit{Closure} & := \lambda P.P\ \mathit{True}\ :\ ((v \to t) \to t) \to t \\
C_{john} & := \lambda P.\ P\,\mathrm{john} \\
C_{walks} & := \lambda adv_s\ adv_v\ subj.\ adv_s\,(subj\,(adv_v(\lambda x.\lambda h.\exists w.\,(\mathrm{walk}\,w) \wedge (arg_0\,w\,x) \wedge (h\,w))) \\
C_{smart} & := \lambda D.\lambda n.\lambda q.\lambda f.D(\lambda xh.(n\,x\,h) \wedge (\mathrm{smart}\,x))q\,f \\
C_{every}^{\mathbf{n_A}} & := \lambda p.\lambda q.\lambda f.\forall x.(p\,x\,f) \supset (q\,x\,f) \\
C_{woman}^{\mathbf{n_A} \multimap \mathbf{NP}} & := \lambda D.D(\lambda\,x\,h.(\mathrm{woman}\,x \wedge h\,x)) \\
C_{certainly}^{\mathbf{S}_A \multimap \mathbf{S}_A} & := \lambda m.\,\lambda V.\ m\,(\lambda h.V(\lambda v.(\mathrm{certainly}\,v) \wedge (h\,v)) \\
C_{fast}^{\mathbf{VP}_A \multimap \mathbf{VP}_A} & := \lambda m.\,\lambda V.\,m\,(\lambda x.\lambda h.\,Vx(\lambda v.(\mathrm{fast}\,v) \wedge (h\,v))) \\
C_{does\ not}^{\mathbf{VP}_A} & := \lambda Vxh.\neg(V\,x\,h)
\end{array}
$$

Second try: No continuations for noun phrases

New interpretations

$$
\begin{aligned}
C_{john} &:= \lambda P.\, P\,\mathrm{john} : (e \to \Omega) \to \Omega \\
C_{walks} &:= \lambda adv_s\; adv_v\; subj\,.adv_s\,(subj\,(adv_v(\lambda x.\lambda h.\exists w.\,(\mathrm{walk}\; w) \wedge (arg_0\; w\; x) \wedge (h\, w)))) \\
C_{woman} &:= \lambda D.D(\lambda\, x.\mathrm{woman}\, x) \\
C_{every} &:= \lambda PQ.\lambda h.\forall x(Px \supset Qxh) : (e \to t) \to (e \to \Omega) \to \Omega \\
C_{a} &:= \lambda PQ.\lambda h.\exists x(Px \wedge Qxh) : (e \to t) \to (e \to \Omega) \to \Omega \\
C_{smart} &:= \lambda D.\lambda n.\lambda q.\lambda f\,.D(\lambda x.(n\, x\,) \wedge (\mathrm{smart}\; x))q\, f \\
C_{certainly} &:= \lambda m.\, \lambda V.\;\; m\,(\lambda h.V(\lambda v.(\mathrm{certainly}\; v) \wedge (h\, v)) \\
C_{fast} &:= \lambda m.\, \lambda V\,.\, m\,(\lambda x.\lambda h.\,Vx(\lambda v.(\mathrm{fast}\; v) \wedge (h\, v))) \\
C_{does\ not} &:= \lambda Vxh.\neg(V\; x\; h) \\
C_{itisnotthecase} &:= \lambda S\; h.\neg(S\; h)
\end{aligned}
$$

Where: $\Omega \equiv_{\mathrm{def}} (v \to t) \to t$

Second try: No continuations for noun phrases

New interpretations

$$
\begin{array}{ll}
C_{john} & := \lambda P.\, P\, \text{john} : (e \to \Omega) \to \Omega \\
C_{walks} & := \lambda adv_s\, adv_v\, subj\, .adv_s\, (subj\, (adv_v(\lambda x.\lambda h.\exists w.\, (\text{walk}\, w) \wedge (arg_0\, w\, x) \wedge (h\, w)))) \\
C_{woman} & := \lambda D.D(\lambda\, x.\text{woman}\, x) \\
C_{every} & := \lambda PQ.\lambda h.\forall x(Px \supset Qxh) : (e \to t) \to (e \to \Omega) \to \Omega \\
C_{a} & := \lambda PQ.\lambda h.\exists x(Px \wedge Qxh) : (e \to t) \to (e \to \Omega) \to \Omega \\
C_{smart} & := \lambda D.\lambda n.\lambda q.\lambda f.D(\lambda x.(n\, x) \wedge (\text{smart}\, x))q\, f \\
C_{certainly} & := \lambda m.\, \lambda V.\ m\, (\lambda h.V(\lambda v.(\text{certainly}\, v) \wedge (h\, v)) \\
C_{fast} & := \lambda m.\, \lambda V.\, m\, (\lambda x.\lambda h.Vx(\lambda v.(\text{fast}\, v) \wedge (h\, v))) \\
C_{does\ not} & := \lambda Vxh.\neg(V\, x\, h) \\
C_{itisnotthecase} & := \lambda S\, h.\neg(S\, h)
\end{array}
$$

Where: $\Omega \equiv_{\text{def}} (v \to t) \to t$

$$M_3 := \forall x(\text{woman}\, x \wedge \text{smart}\, x \supset \exists w(\text{walk}\, w) \wedge (arg_0\, w\, x) \wedge (\text{fast}\, w))$$

$$M_4 := \forall x(\text{woman}\, x \wedge \text{smart}\, x \supset \exists w(\text{walk}\, w) \wedge (arg_0\, w\, x) \wedge (\text{certainly}\, w))$$

Bonus: Coreference, Raising

$$
\begin{array}{lll}
C_{wants}: & \mathbf{S}_A \multimap \mathbf{VP}_A \multimap \mathbf{NP} \multimap \mathbf{S}'_A & C_{to\text{-}sleep}: \quad \mathbf{S}'_A \multimap \mathbf{S} \\
\hline
C_{wants} := & \lambda adv_s\, adv_v\, subj.\lambda Pred.adv_s\,(subj(adv_v.\lambda x\, h. & \\
 & \exists w((\mathrm{want}\, w) \wedge (h\, w) \wedge (arg_0\, w\, x) \wedge Pred(\lambda Q.Q\, x)(\lambda r.\, Arg_1\, w\, r)) & \\
C_{to\text{-}sleep} := & \lambda cont.cont(\lambda subj.subj(\lambda x.\lambda f.\exists u.(\mathrm{sleep}\, u) \wedge (arg_0\, u\, x) \wedge (f\, u)) & \\
\mathbf{S}'_A := & (((e \to \Omega) \to \Omega) \to \Omega) \to \Omega &
\end{array}
$$

(5) a. John wants to sleep.

$M_5 = Closure(C_{to\text{-}sleep}\,(C_{wants}\,\mathrm{I}_{\mathbf{S}}\,\mathrm{I}_{\mathbf{VP}}\,C_{john}))\ :\ \mathbf{T}$

$\exists w(\mathrm{want}\, w) \wedge (arg_0\, w\, \mathrm{john}) \wedge (\exists u(\mathrm{sleep}\, u) \wedge (Arg_1\, w\, u) \wedge (arg_0\, u\, \mathrm{john}))$

b. Every boy wants to sleep.

$M_6 = Closure(C_{to\text{-}sleep}\,(C_{wants}\,\mathrm{I}_{\mathbf{S}}\,\mathrm{I}_{\mathbf{VP}}(C_{boy}\,C_{every})))\ :\ \mathbf{T}$

$\forall x(\mathrm{boy}\, x \supset \exists w(\mathrm{want}\, w) \wedge (arg_0\, w\, x) \wedge (\exists u.(\mathrm{sleep}\, u) \wedge (Arg_1\, w\, u) \wedge (arg_0\, u\, x)))$

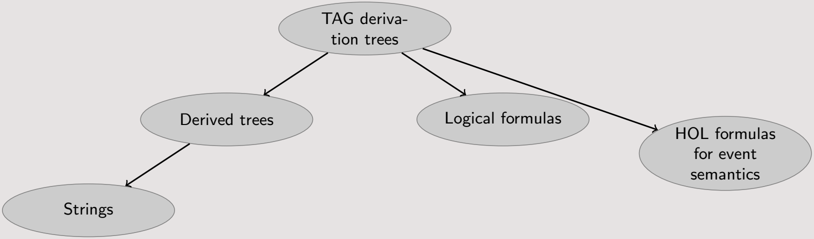
c. Every boy does not want to sleep.

$M_7 = Closure(C_{to\text{-}sleep}\,(C_{wants}\,\mathrm{I}_{\mathbf{S}}\,\mathrm{I}_{\mathbf{VP}}(C_{boy}\,C_{every})))\ :\ \mathbf{T}$

$\forall x(\mathrm{boy}\, x \supset \neg(\exists w(\mathrm{want}\, w) \wedge (arg_0\, w\, x) \wedge (\exists u.(\mathrm{sleep}\, u) \wedge (Arg_1\, w\, u) \wedge (arg_0\, u\, x))))$

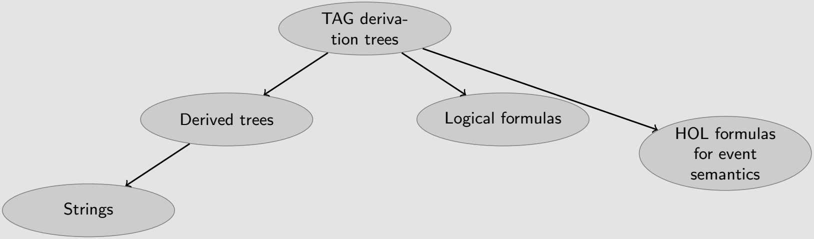
only one reading out of two

Future Work and Conclusion



Current approach

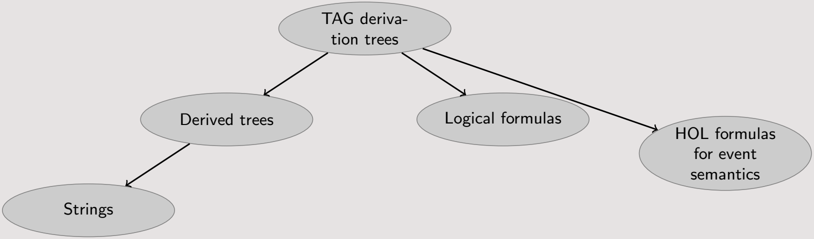
Future Work and Conclusion



Current approach

- TAG derivation trees to Stable's HOL translation of AMRs using ACGs
- Coreference missing in AAMR
- An approach to NLG with HOL encodings of AMRs for free

Future Work and Conclusion



Current approach

- TAG derivation trees to Stable's HOL translation of AMRs using ACGs
- Coreference missing in AAMR
- An approach to NLG with HOL encodings of AMRs for free

Future work

- Encode more complex interaction of quantifiers and negation
- A large scale ACG
- Maintain reasonable bounds on parsing/generation complexity

Thank You

References I

Artzi, Yoav, Kenton Lee, and Luke Zettlemoyer (2015). "Broad-coverage CCG Semantic Parsing with AMR". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1699–1710. DOI: 10.18653/v1/D15-1198. URL: http://www.aclweb.org/anthology/D15-1198.

Banarescu, Laura et al. (2013). "Abstract Meaning Representation for Sembanking". In: *Proceedings of the 7th Linguistics Annotation Workshop & Interoperability with Discourse*. Sofia, Bulgaria, pp. 178–186.

Bos, Johan (2016). "Expressive Power of Abstract Meaning Representations". In: *Computational Linguistics* 42.3, pp. 527–535. DOI: 10.1162/COLI_a_00257. eprint: https://doi.org/10.1162/COLI_a_00257. URL: https://doi.org/10.1162/COLI_a_00257.

Champollion, Lucas (2015). "The interaction of compositional semantics and event semantics". In: *Linguistics and Philosophy* 38.1, pp. 31–66. ISSN: 1573-0549. DOI: 10.1007/s10988-014-9162-8.

References II

De Groote, Philippe (2001). "Towards Abstract Categorial Grammars". In: *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, pp. 148–155. acl: P01-1033. URL: http://aclweb.org/anthology/P/P01/P01-1033.

Flanigan, Jeffrey et al. (2016). "Generation from Abstract Meaning Representation using Tree Transducers". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, pp. 731–739. DOI: 10.18653/v1/N16-1087. URL: http://www.aclweb.org/anthology/N16-1087.

Joshi, Aravind K., Leon S. Levy, and Masako Takahashi (1975). "Tree Adjunct Grammars". In: *Journal of Computer and System Sciences* 10.1, pp. 136–163. DOI: 10.1016/S0022-0000(75)80019-5.

Kanazawa, Makoto (2007). "Parsing and Generation as Datalog Queries". In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*. Prague, Czech Republic: Association for Computational Linguistics, pp. 176–183. acl: P07-1023. URL: http://www.aclweb.org/anthology/P07-1023.

References III

Pogodalla, Sylvain (2004a). "Computing Semantic Representation: Towards ACG Abstract Terms as Derivation Trees". In: *Proceedings of TAG+7*, pp. 64–71. URL: http://hal.inria.fr/inria-00107768/PDF/A04-R-058.pdf.

– (2004b). "Computing Semantic Representation: Towards ACG Abstract Terms as Derivation Trees". In: *Proceedings of the Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+7)*, pp. 64–71.

– (2017). "A syntax-semantics interface for Tree-Adjoining Grammars through Abstract Categorial Grammars". In: *Journal of Language Modelling* 5.3, pp. 527–605. DOI: 10.15398/jlm.v5i3.193. URL: https://hal.inria.fr/hal-01242154.

Salvati, Sylvain (2005). "Problèmes de filtrage et problèmes d'analyse pour les grammaires catógorielles abstraites". PhD thesis. Institut National Polytechnique de Lorraine. URL: http://www.labri.fr/perso/salvati/downloads/articles/these.pdf.

Stabler, Edward (2018). "Reforming AMR". In: *Formal Grammar*. Ed. by Annie Foret, Reinhard Muskens, and Sylvain Pogodalla. Springer Berlin Heidelberg, pp. 72–87. ISBN: 978-3-662-56343-4.

Winter, Yoad and Joost Zwarts (2011). "Event Semantics and Abstract Categorial Grammar". In: *The Mathematics of Language*. Ed. by Makoto Kanazawa et al. Springer Berlin Heidelberg, pp. 174–191.