Computational Semantics

Sylvain Pogodalla[1]

[1] mailto:sylvain.pogodalla@inria.fr
http://www.loria.fr/~pogodall
Office: C. 302
LORIA/INRIA Nancy–Grand Est
France

Lorraine University NLP Master, 2012 – 2013

Outline

Computational Semantics?

What is computational semantics?

- A suitable interpretation level
- A way to relate semantic structures and syntax (interpretation or realization)

Sense and Denotation

Gottlob Frege

- Sense = mode of presentation
- Denotation = the object it refers to

Ex.: $1 + 1$ and 2 have the same denotation but not the same sense

Meaning as an Observable Property [Winter(2010)]

- Select a specific property of language, stable across speakers and situations
- Extra-linguistic and intra-linguistic semantic phenomena

Example (Extra-linguistic)

- What is common to the objects that people categorize as being red?
- What the effect of asking *please pick a blue card from this pack*?

Example (Intra-Linguistic)

- How do speaker identify relations between pairs of words like *red-color*, *dog-animal*?
- What are the relations between the sentences *please pick a blue card from this pack* and *please pick a card from this pack*?

- Contrasts

Example

- Red is a color / ?Red is an animal
- Every red thing has a color / ?Every red thing has an animal

Entailment

Example

- Tina is tall and thin
- Tina is thin

Example

- Tina is tall and thin $\Rightarrow$ Tina is thin
- A dog entered the room $\Rightarrow$ An animal entered the room
- John picked a blue card from this pack $\Rightarrow$ John picked a card from this pack

- Stability of judgments
- Indefeasible reasoning (vs. *Tina is a bird. Tina can fly*)
- Entailment judgements $\equiv$ grammaticality judgments
- Models and denotation

Models and Truth-Conditionality Criterion

Aim

Establishing a relation between language expressions and objects in models.

- Formal semantics
- Applied semantics

Linking *exp* and $[\![exp]\!]^M$ where M is a model.

Denotations of sentences are truth-values 1 (for "true") and 0 (for "false")

Definition (TCC)

A semantic theory T satisfies the truth-conditionality criterion (TCC) if for all sentences S_1 and S_2 the following conditions are equivalent:

1. Sentences S_1 intuitively entails sentence S_2
2. For all models M in T $[\![S_1]\!]^M \leq [\![S_2]\!]^M$

- 1 is an empirical statement
- 2 is a theoretical statement

Models and Truth-Conditionality Criterion (cont'd)

Entailment

$S_1 \Rightarrow S_2$

$\overset{\text{TCC}}{\longleftrightarrow}$

Models

$$[\![S_1]\!]^{M_1} \leq [\![S_2]\!]^{M_1}$$
$$[\![S_1]\!]^{M_2} \leq [\![S_2]\!]^{M_2}$$
$$[\![S_1]\!]^{M_3} \leq [\![S_2]\!]^{M_3}$$
$$\ldots$$

Building Denotations

For every model M:

- In addition to truth-values, there exists E_M an arbitrary non-empty set of entities in M
- *Tina* denotes an arbitrary entity **tina** $= [\![Tina]\!]^M$ in E_M
- *tall* and *thin* denote arbitrary sets **tall** $= [\![tall]\!]^M$ and **thin** $= [\![thin]\!]^M$ of entities in E_M

What does it mean that *Tina is thin* is true?

- $\mathbf{tina} \in \mathbf{thin}$
- $\mathbf{is}(x, A) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$
- $[\![Tina\ is\ thin]\!]^M = \mathbf{is}(\mathbf{tina}, \mathbf{thin})$

Constants

is is constant across models!

Exercice: What's the denotation of *Tina is tall and thin*?

Tina is tall and thin

- $\textbf{and}(A, B) = A \cap B$
- $[\![\textit{Tina is tall and thin}]\!]^M = \textbf{is}(\textbf{tina}, \textbf{and}(\textbf{tall}, \textbf{thin}))$

Denotations

Expressions	Category	Type	Abstract denotation
Tina	*NP*	entity	**tina**
tall	*A*	set of entities	**tall**
thin	*A*	set of entities	**thin**
tall and thin	*AP*	set of entities	$\textbf{and}(\textbf{tall}, \textbf{thin})$
Tina is thin	*S*	truth-value	$\textbf{is}(\textbf{tina})$
Tina is tall and thin	*S*	truth-value	$\textbf{is}(\textbf{tina}, \textbf{and}(\textbf{tall}, \textbf{thin}))$

Example

- Show that *Tina is tall and thin* $\Rightarrow$ *Tina is thin*
- Show that *Tina is thin* $\not\Rightarrow$ *Tina is tall and thin*

Involving Syntactic Structures

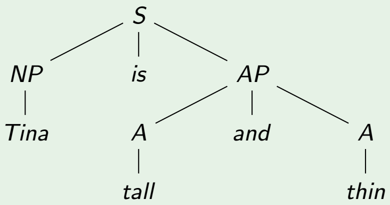
Example

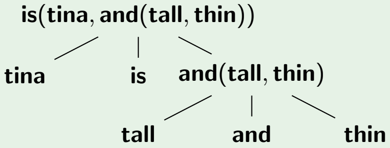
- All pianists are composers and Tina is a pianist.
- All composers are pianists and Tina is a pianist.
- $\stackrel{?}{\Rightarrow}$ Tina is a composer.

Compositionality

The denotation of a complex expression is determined by the denotations of its immediate parts and the ways they combine with each other.

Example





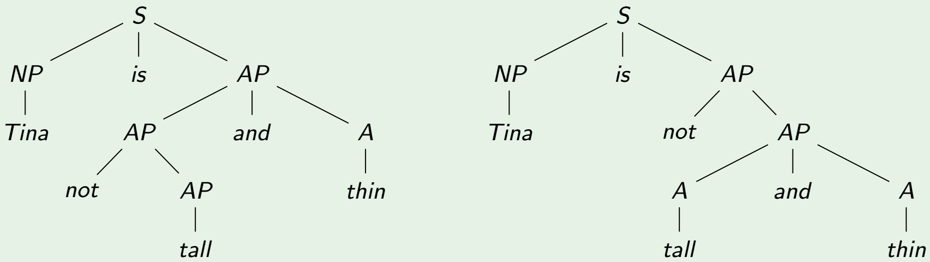
About Compositionality

- Direct compositionality
- Some syntactic assumptions
- How the denotations of functions know what their arguments are?

Structural Ambiguity

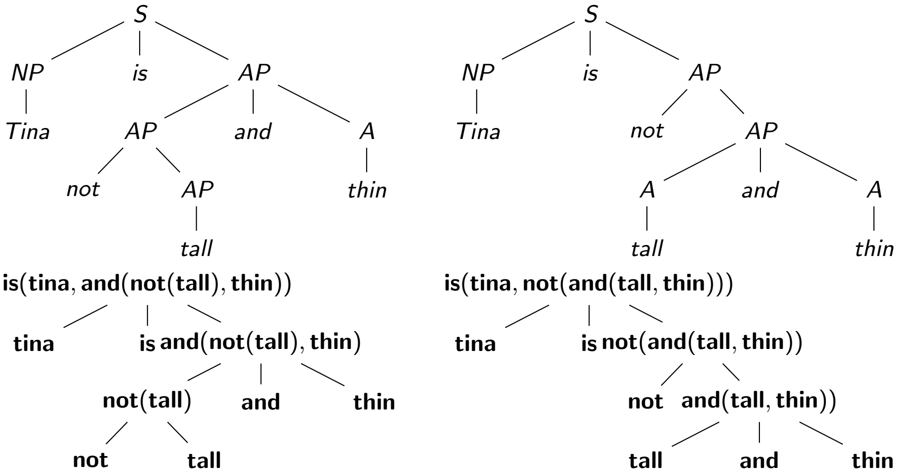
Example (Tina is not tall and thin)

Tina is not tall and thin $\stackrel{?}{\Rightarrow}$ *Tina is thin*



- What makes the syntactic ambiguity a semantic ambiguity? Compositionality
- What's the denotation of *not*?
- What are the denotations of the two structures?

Structural Ambiguity (cont's)



Exercice [Winter(2010)] I

Two sentences are called *equivalent* if they entail each other as in the following example:

- *Tina is tall and thin* ⇔ *Tina is both tall and thin*

1. Give more examples for pairs of sentences that you consider intuitively for equivalent sentences
2. State the condition that the TCC requires for equivalent sentences
3. Assuming the structure [*both*[*tall and thin*]], how can we define $[\![both]\!]^M$? Is it a constant?
4. Consider the ungrammaticality of the following strings:
    - * *Tina is both tall*
    - * *Tina is both not tall*
    - * *Tina is both tall or thin*

    Let's assume that *both* only appears in adjective phrases as adjacent to *and* conjunctions.

    1. Analyze the truth-values assigned to:
        - *Tina is both not tall and thin*
        - *Tina is not both tall and thin*
    2. Show that this accounts for:
        - *Tina is both not tall and thin* ⇒ *Tina is thin*
        - *Tina is not both tall and thin* ⇏ *Tina is thin*

Exercice [Winter(2010)] II

5. Consider the following sentences:
   - *Tina is not tall and not thin*
   - *Tina is neither tall nor thin*

   Assume that $[\![neither]\!]^M = [\![both]\!]^M$? What should then the denotation of *nor* be to have these two sentences equivalent?

Exercice [Winter(2010)]

Two sentences are called *contradictory* in a given theory if whenever one of them denotes 1 in the theory, the other denotes 0 (*e.g. Mary is not tall* and *Mary is tall*). We may also talk about two contradictory readings/structures of sentences, which is especially useful when sentences are structurally ambiguous.

1. Give more examples for contradictory sentences/structures
2. Consider the sentences *The bottle is empty* and *The bottle is full*. Can you think o a theoretical assumption that would render these sentences contridactory?
3. Give an entailment using this assumption

Types and Domains

Denotations

Expressions	Category	Type	Abstract denotation
Tina	*NP*	entity	**tina**
tall	*A*	set of entities	**tall**
tall and thin	*AP*	set of entities	**and**(**tall**, **thin**)
Tina is thin	*S*	truth-value	**is**(**tina**,**thin**)
is	?	function from entities and set of entities to truth-values	**is**
and	?	function from pairs of set of entities to set of entities	**and**
not	?	function from set of entities to set of entities	**not**

- Systematic relation between expressions of a given syntactic category to a type of denotation
- Distinction between the objects of the model (entities, set of entities, functions, etc.)

⇒ Domains: parts of the model that gathers objects with the same structure

- The property for objects to belong to the same domain is expressed by having same type

Types and Domains (cont'd)

Definition (Characteristic Function)

Let A be a subset of B. A function F_A from B to $\{0,1\}$ the set of truth-values is called the *characteristic function of* A *in* B if it satisfies for every $x \in B$:

$$f_A(x) = \begin{cases} 1 \text{ if } x \in A \\ 0 \text{ if } x \notin A \end{cases}$$

Types and Domains

Basic types	
Type	Domain
e	$E = D_e$
t	$\{0,1\} = D_t$

Complex types	
Type	Domain
$e \rightarrow t$	$D_t^{D_e}$
...	...

Example

Let M be a model such that $D_e = \{\mathbf{t}, \mathbf{j}, \mathbf{m}\}$. How many possible denotations for *tall* are there?

Types

Definition (Types)

The set of types $\mathscr{T}$ over the basic types e and t is the smallest set such that:

- $\{e, t\} \subseteq \mathscr{T}$
- If $\tau \in \mathscr{T}$ and $\sigma \in \mathscr{T}$ then $\tau \rightarrow \sigma \in \mathscr{T}$

Example

- What's the type of an adjective?
- What's the type of **not**?
- What's the type of **is**?

Two-Place Relation

One-Place Predicate

Tina smiled is true whenever **tina** $\in$ **smiled**
whenever **smiled**(**tina**) $= 1$

smiled $\in F_{e\to t} = D_t^{D_e}$ or **smiled** $: e \to t$

Two-Place Relation

Tina praised Mary is true whenever **tina** belongs to the set of entities that *praised Mary*
whenever $[\![$*praised Mary*$]\!]$(**tina**) $= 1$

$[\![\textit{praised Mary}]\!] \in D_t^{D_e}$ or $[\![\textit{praised Mary}]\!] : e \to t$

$[\![\textit{praised}]\!] \in {D_t^{D_e}}^{D_e}$ or $[\![\textit{praised}]\!] : e \to e \to t$

Note

- $[\![\textit{Tina [praised Mary]}]\!] = (\textbf{praised}(\textbf{mary}))(\textbf{tina})$
- Characteristic function of a binary relation R: $(f_R(y))(x) = 1$ iff $\langle x, y\rangle \in R$
- Currying: $f : A \times B \to C$ and $g : A \to B \to C$ such that $f(a, b) = (g(a))(b)$

Lexicon [Winter(2010)]

Definition (Frame)

Let $E = D_e$ a set of entities. We define the frame $\mathcal{F}^E$ as:

$$\mathcal{F}^E \triangleq \cup_{\tau \in \mathscr{T}} D_\tau$$

Definition (Lexicon)

Let Σ be a finite vocabulary. A lexical typing function $T_{\mathscr{L}}$ of Σ is any function from Σ to $\mathscr{T}$.

Given a lexical typing function $T_{\mathscr{L}}$, a corresponding lexical interpretation function over Σ and a non-empty set of entities E is any function $I_{\mathscr{L}}$ from Σ to $\mathcal{F}^E$ such that:

$$\forall w \in \Sigma \quad I_{\mathscr{L}}(w) : T_{\mathscr{L}}(w)$$

Definition (Model)

Let Σ be a finite vocabulary with $T_{\mathscr{L}}$ a lexical typing function over Σ. A model over Σ is a pair $\langle E, I_{\mathscr{L}} \rangle$ where E is a non-empty set of entities and $I_{\mathscr{L}}$ is a lexical interpretation function over Σ and E.

Example

Vocabulary	$\Sigma = \{Tina, Mary, smiled, praised\}$
Typing	$T_{\mathscr{L}} : \Sigma \longrightarrow \mathscr{T}$ $Tina \longrightarrow e$ $Mary \longrightarrow e$ $smiled \longrightarrow e \rightarrow t$ $praised \longrightarrow e \rightarrow e \rightarrow t$
Interpretation	$I_{\mathscr{L}} : \Sigma \longrightarrow \mathcal{F}^E$ $Tina \longrightarrow \mathbf{tina}$ $Mary \longrightarrow \mathbf{mary}$ $smiled \longrightarrow \begin{cases} \mathbf{tina} \longrightarrow 1 \\ \mathbf{mary} \longrightarrow 0 \end{cases}$ $praised \longrightarrow \begin{cases} \mathbf{tina} \longrightarrow \begin{cases} \mathbf{tina} \longrightarrow 1 \\ \mathbf{mary} \longrightarrow 1 \end{cases} \\ \mathbf{mary} \longrightarrow \begin{cases} \mathbf{tina} \longrightarrow 1 \\ \mathbf{mary} \longrightarrow 0 \end{cases} \end{cases}$
Lexicon	$\langle \Sigma, T_{\mathscr{L}} \rangle$
Model	$\langle E, I_{\mathscr{L}} \rangle$

- What's the denotation of *Mary smiled*?
- What's the denotation of *Mary praised Mary*?

About the Lexicon

Definition (Restricting Functor)

Let Σ be a finite vocabulary and $T_{\mathscr{L}}$ be a lexical typing function from Σ to $\mathscr{T}$. Let E be a non-empty set.
A restricting functor $\mathcal{RF}^E$ over Σ is a function that maps any word $w \in \Sigma$ to a subset of the domain $D_{T_{\mathscr{L}}(w)}$.
For $w \in \Sigma$:

- if $\mathcal{RF}^E(w) = D_{T_{\mathscr{L}}(w)}$ for every set E, we say that the denotation of w is arbitrary
- if $\mathcal{RF}^E(w)$ is a singleton for every set E, we say that the denotation of w is constant

- Have we seen *constants*?
- What about “intermediary” kinds of lexical items?
  - $\mathcal{SYM}^E$ is the set of symmetric functions in $D_{e \to e \to t}$ (f is symmetric iff $f(x)(y) = f(y)(x)$)
  - $\mathcal{RMOD}^E$ is the set of restrictive modifiers in $D_{(e \to t) \to (e \to t)}$ where $f \in \mathcal{RMOD}^E$ iff $\forall g \in D_{e \to t}\, \forall x \in D_e$, if $(f(g))(x) = 1$ then $g(x) = 1$ as well
- Do you have examples?
  - *resemble*
  - *very*
  - *charmingly*

Theory and Models

Definition (Intended Model)

Let Σ be a finite vocabulary with $T_{\mathscr{L}}$ a lexical typing function over Σ. Let E be a non-empty set and $\mathcal{RF}^E$ a restricting functor over Σ. A model $\langle E, I_{\mathscr{L}} \rangle$ over Σ is an intended model if for every word $w \in \Sigma$ $I_{\mathscr{L}}(w) \in \mathcal{RF}^E(w)$.

Definition (Truth-Conditionality Criterion)

A semantic theory T *that specifies a typing function* $T_{\mathscr{L}}$ *and a restricting functor* $\mathcal{RF}^E$ *over* Σ satisfies the truth-conditionality criterion (TCC) if for all *structures* S_1 and S_2 the following conditions are equivalent:

1. *Structure* S_1 intuitively entails *structure* S_2
2. For all *intended* models M in T $[\![S_1]\!]^M \leq [\![S_2]\!]^M$

Reminder

Definition (λ-Calculus)

Syntax $\mathcal{V} = \{x, y, \ldots\}$ and $T ::= \mathcal{V} | \lambda\mathcal{V}.T | (T\ T)$

Free Variables Let $t \in T$

- $\mathrm{FV}(x) = \{x\}$
- $\mathrm{FV}(\lambda x.u) = \mathrm{FV}(u) \setminus \{x\}$
- $\mathrm{FV}(t\ u) = \mathrm{FV}(t) \cup \mathrm{FV}(u)$

If $\mathrm{FV}(t) = \emptyset$ t is closed.

Example

What are the free variables of

- $x\ y\ z$
- $\lambda x.x\ y$
- $(\lambda x.x\ x)(\lambda y.y\ y)$

Reminder (cont'd)

Definition (Substitution)

For $t, u \in T$ and $x \in \mathcal{V}$, the substitution of u for x in t, written $t[x := u] \in T$, is defined as follows ($x \neq y$):

- $x[x := u] = u$
- $y[x := u] = y$
- $(v\, w)[x := u] = v[x := u]\, w[x := u]$
- $(\lambda x.v)[x := u] = \lambda x.v$
- $(\lambda y.v)[x := u] = \lambda y.v[x := u]$ with $y \notin \mathrm{FV}(u)$

Example

Compute $((\lambda x.x\, y\, z)(\lambda y.x\, y\, z)(\lambda z.x\, y\, z))[x := y]$

Reminder (cont'd)

Definition (Reduction)

Reduction $(\lambda x.t)\, u \rightarrow_\beta t[x := u]$

Church-Rosser Theorem For all λ-terms t, u and v such that

$$t \rightarrow_\beta^* u \text{ and } t \rightarrow_\beta^* v$$

there exists w such that

$$u \rightarrow_\beta^* w \text{ and } v \rightarrow_\beta^* w$$

Example

Let $\delta = \lambda x.x\, x$. Reduce $\Omega = \delta\, \delta$.

Reminder (cont'd)

Definition (Simply Typed λ-Calculus)

$\Gamma = x_1 : A_1, \ldots, x_n : A_n$ is a context.

$$\frac{}{\Gamma, x : A \vdash x : A}\ \text{Axiom}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ \text{Abs.} \qquad \frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash (t\, u) : B}\ \text{App.}$$

Example

- What about the types of $\lambda x.x$? Of $\lambda xy.x$? Of $\lambda x.\lambda y.\lambda z.x\, z\, (y\, z)$?
- Can you type $\delta = \lambda x.x\, x$?

Syntactic Structures

Definition (Binary Structure)

Given a vocabulary Σ, a binary structure over Σ is one of the following:

1. An occurrence of a word $w \in \Sigma$.
2. A sequence $[S_1 \quad S_2]$ where S_1 and S_2 are binary structures over Σ.

Definition (Denotation of a Structure)

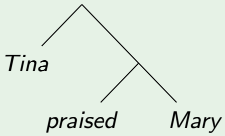
Let Σ be a vocabulary, E be a non-empty set of entities, $T_{\mathscr{L}}$ be a lexical typing function over Σ and $I_{\mathscr{L}}$ be a corresponding lexical denotation function over Σ. Then for every binary structure S over Σ, the syntactic typing and denotation functions T_{s} and I_{s} extend $T_{\mathscr{L}}$ annd $I_{\mathscr{L}}$ as follows:

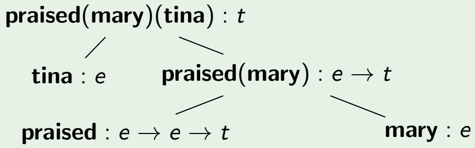
- $$T_{\mathrm{s}}(S) = \begin{cases} T_L(w) & \text{if } S \text{ is a word } w \in \Sigma \\ \beta & \text{if } S = [S_1 \quad S_2] \text{ and } T_{\mathrm{s}}(S_1) = \alpha \to \beta \text{ and } T_{\mathrm{s}}(S_2) = \alpha \\ \beta & \text{if } S = [S_1 \quad S_2] \text{ and } T_{\mathrm{s}}(S_1) = \alpha \text{ and } T_{\mathrm{s}}(S_2) = \alpha \to \beta \\ \text{undefined} & \text{otherwise} \end{cases}$$
- $$I_{\mathrm{s}}(S) = \begin{cases} I_L(w) & \text{if } S \text{ is a word } w \in \Sigma \\ (t\ u) & \text{if } S = [S_1 \quad S_2] \text{ and } I_{\mathrm{s}}(S_1) = t : \alpha \to \beta \text{ and } I_{\mathrm{s}}(S_2) = u : \alpha \\ (t\ u) & \text{if } S = [S_1 \quad S_2] \text{ and } I_{\mathrm{s}}(S_1) = u : \alpha \text{ and } I_{\mathrm{s}}(S_2) = t : \alpha \to \beta \\ \text{undefined} & \text{otherwise} \end{cases}$$

Examples

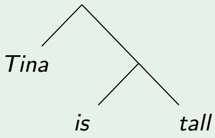
Example (*Tina praised Mary*)

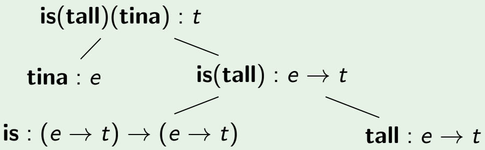
[*Tina*[*praised Mary*]]





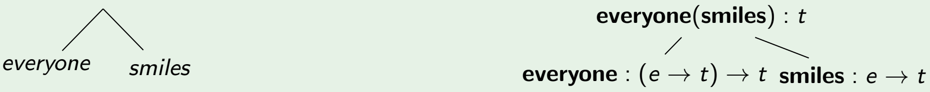
Example (*Tina is tall*)



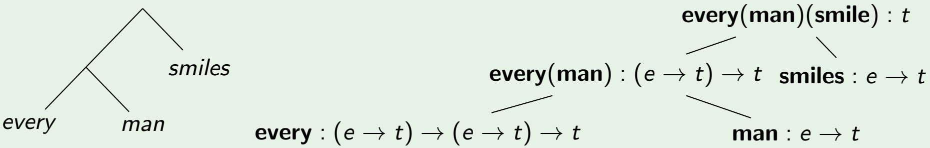


Quantification

Example (*Everyone smiles*)



Example (*Every man smiles*)



- *Shake and bake* semantics
- What's the denotation of *everyone*?
- What's the denotation of *every*?

Computing Denotations with Functions or Logical Formulas

So far

tall as:

- Set
- (Characteristic) Function
- Symbol of relation

⇒ Move to logic

What kind of logic?

Higher-Order Logic

HOL

- Two atomic types: e and t (or ι and o)
- Logical constants:

$$\begin{array}{ll} \bot : & t \\ \Rightarrow : & t \to t \to t \\ \forall_\alpha : & (\alpha \to t) \to t \text{ for each type } \alpha \end{array}$$

- Formulas: well-typed terms of type t

- Contrast with first order logic
- Build a HOL formula which is not FOL

Motivations

There is in my opinion no important theoretical difference between natural languages and the artificial languages of logicians; indeed, I consider it possible to comprehend the syntax and semantics of both kinds of languages within a single natural and mathematically precise theory. On this point I differ from a number of philosophers (...). [Montague(1970)]

Montague Semantics [Montague(1974)]

The aim of this paper is to present in a rigorous way the syntax and semantics of a certain fragment of a certain dialect of English. [*Montague(1974)*]

- Fragment
- Semantic types as homomorphic image of syntactic types
- Semantic representation as translation of syntactic operations
- Semantic representation through a "*certain artificial language*", a logical language

The Original Version

Categories of English

- Basic types: e and t
- Type constructors: A/B and $A//B$
- Some definitions:
    - IV, or the category of instransitive verb phrases, is to be t/e.
    - T, or the category of terms, is to be $t/$IV,
    - TV, or the category of transitive verb phrases, is to be IV/T.
    - CN, or the category of common noun phrases, is to be $t//e$.
    - ...
- B_A the set of basic expressions of the category A. P_a is the set of *phrases* of the category A.
    - $love \in B_{\mathrm{TV}}$
    - $Mary \in B_{\mathrm{T}}$, $he_0 \in B_{\mathrm{T}}$
    - $man \in B_{\mathrm{CN}}$

Syntactic Rules

Basic Rules

S1 $B_A \subset P_A$ for every category A.

S2 If $\zeta \in P_{\mathrm{CN}}$, then $F_0(\zeta), F_1(\zeta), F_2(\zeta) \in P_{\mathrm{T}}$ where:

- $F_0(\zeta) =$ *every* ζ
- $F_1(\zeta) =$ *the* ζ
- $F_2(\zeta)$ is *a* ζ or *an* ζ according as the first word in ζ takes *a* or *an*

...

Syntactic Rules

Rules of functional application

S4 If $\alpha \in P_{t/\mathrm{IV}}$ and $\delta \in P_{\mathrm{IV}}$, then $F_4(\alpha, \delta) \in P_t$, where $F_4(\alpha, \delta) = \alpha\delta'$ and δ' is the result of replacing the first *verb* in δ by its third person singular present

S5 If $\delta \in P_{\mathrm{IV}/\mathrm{T}}$ and $\beta \in P_{\mathrm{T}}$, then $F_5(\delta, \beta) \in P_{\mathrm{IV}}$, where $F_5(\delta, \beta) = \delta\beta$ if β does not have the form he_n and $F_5(\delta, he_n) = \delta\ him_n$

...

Syntactic Rules

Rules of quantification

S14 If $\alpha \in P_{\mathrm{T}}$ and $\phi \in P_t$, then $F_{10,n}(\alpha, \phi) \in P_t$, where either:

1. α does not have the form he_k, and $F_{10,n}(\alpha, \phi)$ comes from ϕ by replacing the first occurence of he_n or him_n by α and all other occurrences of he_n or him_n by $\left\{\begin{array}{c} he \\ she \\ she \end{array}\right\}$ or $\left\{\begin{array}{c} him \\ her \\ it \end{array}\right\}$ respectively, according as the gender of the first B_{CN} or B_{T} in α is $\left\{\begin{array}{l} \text{masc.} \\ \text{fem.} \\ \text{neuter} \end{array}\right\}$, or
2. $\alpha = he_k$, and $F_{10,n}(\alpha, \phi)$ comes from ϕ by replacing all occurrences of he_n or him_n by he_k or him_k respectively

...

Every man loves Mary

every man loves Mary $(t, \text{S14}, F_{10,0}(\textit{every man}, \textit{he}_0 \textit{ loves Mary}))$

- *every man* $(\textsc{t}, \text{S2}, F_0(\textit{man}))$
  - *man* $(\textsc{cn})$
- $\textit{he}_0$ *loves Mary* $(t, \text{S4}, F_4(\textit{he}_0, \textit{love Mary}))$
  - $\textit{he}_0$ $(\textsc{t} = t/\textsc{iv})$
  - *love Mary* $(\textsc{iv}, \text{S5}, F_5(\textit{love}, \textit{Mary}))$
    - *love* $(\textsc{tv} = \textsc{iv}/\textsc{t})$
    - *Mary* $(\textsc{t})$

Translating English into (Intensional) Logic

Categories to Semantic Types

f is a function such that

- $f(e) = e$
- $f(t) = t$
- $f(A/B) = f(A//B) = f(B) \to f(A)$ where A, B are categories

Translation rules: the $\overline{\cdot}$ function

T1 If α is in the domain of g, then $\overline{\alpha} = g(\alpha)$ [interpretation of constants].
$\overline{he_n} = \lambda P.P\, x_n \ldots$

T2 if $\zeta \in P_{\mathrm{CN}}$ then $\overline{every\ \zeta} = \lambda P.\forall x.\overline{\zeta}(x) \Rightarrow P(x), \ldots$

...

T4 if $\delta \in P_{t/\mathrm{IV}}$, $\beta \in P_{\mathrm{IV}}$ then $\overline{F_4(\delta, \beta)} = \overline{\delta}(\overline{\beta})$

T5 if $\delta \in P_{\mathrm{IV/T}}$, $\beta \in P_{\mathrm{T}}$ then $\overline{F_5(\delta, \beta)} = \overline{\delta}(\overline{\beta})$

...

T14 If $\alpha \in P_{\mathrm{T}}$, $\phi \in P_t$ then $\overline{F_{10,n}(\alpha, \phi)} = \overline{\alpha}(\lambda x_n.\overline{\phi})$

...

Every man loves Mary

every man loves Mary $(t, \text{S14}, F_{10,0}(\textit{every man}, \textit{he}_0 \textit{ loves Mary}))$

every man $(\textsc{t}, \text{S2}, F_0(\textit{man}))$

man $(\textsc{cn})$
MAN

he$_0$ *loves Mary* $(t, \text{S4}, F_4(\textit{he}_0, \textit{love Mary}))$

he$_0$ $(\textsc{t} = t/\textsc{iv})$
$\lambda P.P\, x_0$

love Mary $(\textsc{iv}, \text{S5}, F_5(\textit{love}, \textit{Mary}))$

love $(\textsc{tv} = \textsc{iv}/\textsc{t})$
$\lambda o x.o\ (\lambda y.\text{LOVE}(x, y))$

Mary $(\textsc{t})$
$\lambda P.P$ MARY

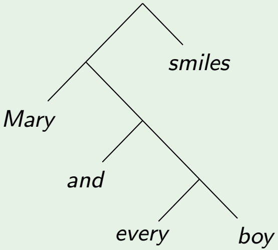
Remarks

- Type homomorphism
- Translation

$\Rightarrow$ What about widespread syntactic formalisms?

Conjunction I

Example (*Mary and every boy smiles*)



Conjunction II

Example (*Mary and every boy smiles*)

(and (every boy) (Mary)) smiles : t
$(\forall_e x.(\textsc{boy}\,x \Rightarrow \textsc{smile}\,x)) \wedge (\textsc{smile}\,\textsc{m})$

and (every boy) (mary) : $(e \rightarrow t) \rightarrow t$
$\lambda r.(\forall_e x.(\textsc{boy}\,x \Rightarrow r\,x)) \wedge ((\lambda P.P\,\textsc{m}))\,r)$
$\rightarrow_\beta \lambda r.(\forall_e x.(\textsc{boy}\,x \Rightarrow r\,x)) \wedge (r\,\textsc{m})$

smiles : $e \rightarrow t$
$\textsc{smile}$

mary : $(e \rightarrow t) \rightarrow t$
$\lambda P.P\,\textsc{m}$

and (every boy) : $((e \rightarrow t) \rightarrow t) \rightarrow (e \rightarrow t) \rightarrow t$
$\lambda Q.\lambda r.((\lambda Q.\forall_e x.(\textsc{boy}\,x \Rightarrow Q\,x))\,r) \wedge (Q\,r)$
$\rightarrow_\beta \lambda Q.\lambda r.(\forall_e x.(\textsc{boy}\,x \Rightarrow r\,x)) \wedge (Q\,r)$

and :
$((e \rightarrow t) \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t) \rightarrow (e \rightarrow t) \rightarrow t$
$\lambda PQ.\lambda r.(P\,r) \wedge (Q\,r)$

every boy : $(e \rightarrow t) \rightarrow t$
$\lambda Q.\forall_e x.(\textsc{boy}\,x \Rightarrow Q\,x)$

every : $(e \rightarrow t) \rightarrow (e \rightarrow t) \rightarrow t$
$\lambda PQ.\forall_e x.(P\,x \Rightarrow Q\,x)$

boy : $e \rightarrow t$
$\textsc{boy}$

Quantification and Object Position I

How can we have both the *NP* subject and the *NP* object be arguments of a a transitive verb?

- Allow for *abstraction* (see later)
- Change the denotation of transitive verbs:
  - $[\![e]\!] = e$
  - $[\![\text{IV}]\!] = t/e$
  - $[\![\text{T}]\!] = (e \rightarrow t) \rightarrow t$
  - $\text{TV} = \text{IV}/\text{T}$
  - $[\![\text{TV}]\!] = ((e \rightarrow t) \rightarrow t) \rightarrow e \rightarrow t$
  - $\text{IV}' = t/\text{T}$
  - $\text{TV}' = \text{IV}/\text{T} = (t/\text{T})/\text{T}$
  - $[\![\text{TV}']\!] = ((e \rightarrow t) \rightarrow t) \rightarrow ((e \rightarrow t) \rightarrow t) \rightarrow t$
  - $[\![\textit{smiles}]\!] = \lambda s.s(\lambda x.\text{SMILE}\, x)$
  - $[\![\textit{loves}]\!] = \lambda os.s(\lambda x.o(\lambda y.\text{LOVE}\, x\, y))$

Quantification and Object Position II

CFG Based Approach

S	$\rightarrow$	$NP \quad VP$	$[\![S]\!]$	$= [\![VP]\!]\,[\![NP]\!]$
VP	$\rightarrow$	$tV \quad NP$	$[\![VP]\!]$	$= [\![tV]\!]\,[\![NP]\!]$
NP	$\rightarrow$	$Det \quad N$	$[\![NP]\!]$	$= [\![Det]\!]\,[\![N]\!]$
Det	$\rightarrow$	a	$[\![Det]\!]$	$= \lambda PQ.\exists x.P\,x \wedge Q\,x$
Det	$\rightarrow$	$every$	$[\![Det]\!]$	$= \lambda PQ.\forall x.P\,x \Rightarrow Q\,x$
tV	$\rightarrow$	$loves$	$[\![tV]\!]$	$= \lambda os.s(\lambda x.o(\lambda y.\textsc{love}\,x\,y))$
N	$\rightarrow$	man	$[\![N]\!]$	$= \textsc{man}$
N	$\rightarrow$	$woman$	$[\![N]\!]$	$= \textsc{woman}$
NP	$\rightarrow$	$Mary$	$[\![NP]\!]$	$= \lambda P.P\ \textsc{m}$
NP	$\rightarrow$	$John$	$[\![NP]\!]$	$= \lambda P.P\ \textsc{j}$
NP	$\rightarrow$	$NP_1 \quad Conj \quad NP_2$	$[\![NP]\!]$	$= \lambda r.[\![Conj]\!]\,([\![NP_1]\!]\,r)\,([\![NP_2]\!]\,r)$
$Conj$	$\rightarrow$	and	$[\![Conj]\!]$	$= \lambda s_1 s_2.s_1 \wedge s_2$

- $[\![\textit{John loves a woman}]\!] =?$
- $[\![\textit{Every man loves some woman}]\!] =?$
- How do you get an object wide scope reading?

Adjectives

Grammar

S	$\rightarrow$ NP VP	$[\![S]\!]$	$= [\![VP]\!]\ [\![NP]\!]$
NP	$\rightarrow$ Det N	$[\![NP]\!]$	$= [\![Det]\!]\ [\![N]\!]$
Det	$\rightarrow$ a	$[\![Det]\!]$	$= \lambda PQ.\exists x.P\,x \wedge Q\,x$
Det	$\rightarrow$ $every$	$[\![Det]\!]$	$= \lambda PQ.\forall x.P\,x \Rightarrow Q\,x$
VP	$\rightarrow$ $smiles$	$[\![VP]\!]$	$= \lambda s.s(\lambda x.\textsc{smile}\,x)$
N	$\rightarrow$ man	$[\![N]\!]$	$= \textsc{man}$
NP	$\rightarrow$ $Mary$	$[\![NP]\!]$	$= \lambda P.P\ \textsc{m}$
NP	$\rightarrow$ $John$	$[\![NP]\!]$	$= \lambda P.P\ \textsc{j}$
N	$\rightarrow$ Adj N	$[\![N]\!]$	$= \lambda x.([\![Adj]\!][\![N]\!])\,x$
Adj	$\rightarrow$ big	$[\![Adj]\!]$	$= \lambda N.\lambda x.\textsc{big}\,x \wedge N\,x$

- $[\![big\ man]\!] = \lambda x.(\textsc{big}\,x) \wedge (\textsc{man}\,x)$ intersective adjectives
- $[\![A\ big\ man\ smiles]\!] =?$
- $[\![beautiful\ dancer]\!] =?$ subsective adjectives
- $[\![former\ student]\!] =?$ non-intersective and non-subsective adjectives

Some Observations on Various Grammatical Formalisms

Syntactic Objects (trees, proofs, f-structures) are somehow prior and semantics must be parasitic on those syntactic objects

[Muskens(2001)]

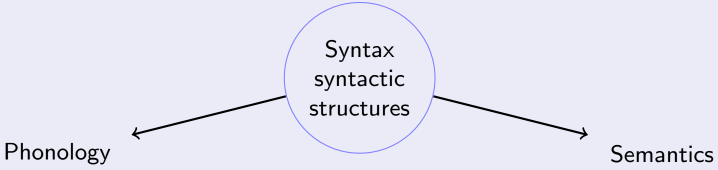
Changing the syntactic analysis to simplify one mapping makes the other mapping more complex. A third possibility is to keep both correspondences simple by localizing the complexity in the syntactic component itself.(...) [T]here is a mismatch between phonology and meaning, which has to be encoded somewhere in the mapping among the levels of structure. If this mismatch is eliminated at one point in the system, it pops up elsewhere.

[Jackendoff(2002), p.15]

Mainstream Architectures

On the Place of the Syntactic Component

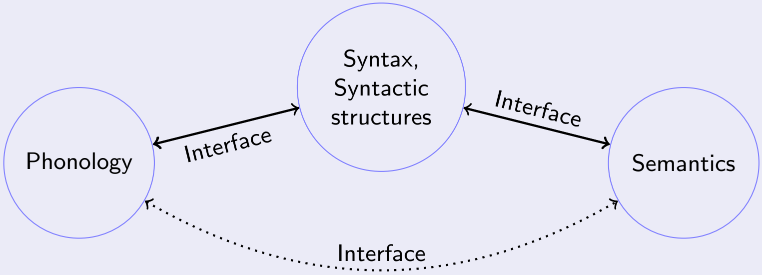
Three Components



- Generative theory: "Free combinatoriality of language is due to a single source, localized in syntactic structure"
- Syntactocentric formalisms = function from the syntactic component to the other ones

A Tripartite Parallel Architecture

Three Components



Weakly Syntactocentric formalisms = relation between the syntactic component and the other ones

Language comprises a number of independent combinatorial systems which are aligned with each other by means of a collection of interface systems. Syntax is among the combinatorial systems, but far from the only one. [Jackendoff(2002)]

λ-terms in the Syntax... and Everywhere

What are λ-terms useful for?

- Montague-like semantics
- Generalization of trees and strings
- Any kind of signatures (atomic types and typed constants): FOL propositions, descriptions (LFG f-structures, URL), other logics
- Very well studied generative system
- Variable binding system

Not that New in Syntax

- [Ranta(1994)],[Oehrle(1994), Oehrle(1995)], [Muskens(2001)], [Muskens(2003)], [Kracht(2003)], [Pollard(2004)], [Pollard(2008)]. . .
- Movements in GB/MG to get S-structures.
- Index Transfer syntactic rule in Binding Theory
- TAG $\rightarrow$ MCTAG

The Tectogrammatical and Phenogrammatical Distinction

On the Grammar Architecture [Curry(1961)]

- Tectogrammatical: abstract combinatorial structure of the grammar
- Phenogrammatical: concrete operations on syntactic data structures (strings, trees, descriptions)
- Contrary to the view that:
  - Syntactic objects are the main objects
  - Semantics (and phonology, and . . .) are by-products

Related Works

[Montague(1974)], [Dowty(1982)],[Ranta(1994)],[Oehrle(1994), Oehrle(1995)], [Muskens(2001)], [Muskens(2003)], [Kracht(2003)], [Pollard(2004)], [Pollard(2008)]. . .

ACG: a Grammatical Framework

Main Features

- ACG is 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* (as in syntactic structures)
  Object language: *Realizations* of the admissible structures
- Both languages are the same objects: sets of (linear) λ-terms

ACG: Formal Properties

Generative Power [de Groote and Pogodalla(2004), Salvati(2006), Kanazawa and Salvati(2007), Kanazawa(2009)]

	String language	Tree language
$\mathrm{ACG}_{(1,n)}$	finite	finite
$\mathrm{ACG}_{(2,1)}$	regular	regular
$\mathrm{ACG}_{(2,2)}$	context-free	linear contex-free
$\mathrm{ACG}_{(2,3)}$	non-duplicating macro well-nested multiple context-free	$\subset$ 1-visit attribute grammar
$\mathrm{ACG}_{(2,4)}$	mildly context-sensitive (multiple context-free)	hyperedge replacement gram.
$\mathrm{ACG}_{(2,4+n)}$	$\mathrm{ACG}_{(2,4)}$	$\mathrm{ACG}_{(2,4)}$
$\mathrm{ACG}_{(3,n)}$	MELL decidability	MELL decidability

Complexity

- $\mathrm{ACG}_{(2,n)}$ parsing is polynomial, equivalent to datalog querying [Salvati(2007), Kanazawa(2007)]
- Reduces to best cases with standard techniques (magic set rewriting) with correct prefix Earley algorithms [Kanazawa(2008)]

ACG Definition $\mathscr{G} = \langle \Sigma_a, \Sigma_o, \mathscr{L}, S \rangle$

Σ_a

$NP, S :$ $type$

$\textsc{Chris} :$ NP

$\textsc{met} :$ $NP \multimap NP \multimap S$

$\mathscr{L}(NP) = \sigma$

$\mathscr{L}(S) = \sigma$

$\mathscr{L}(\textsc{Chris}) = Chris$

$\mathscr{L}(\textsc{met}) = \lambda os.s + met + o$

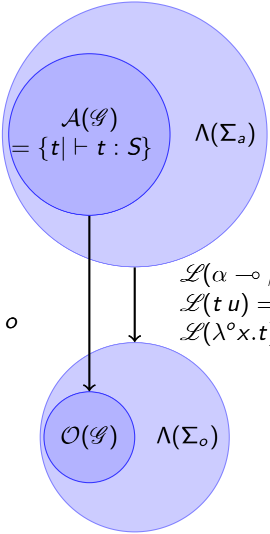
Σ_o

$\sigma :$ $type$

$Chris :$ σ

$met :$ σ

$+ :$ $\sigma \multimap \sigma \multimap \sigma$



$\mathscr{L}(\alpha \multimap \beta) = \mathscr{L}(\alpha) \multimap \mathscr{L}(\beta)$

$\mathscr{L}(t\ u) = \mathscr{L}(t)\ \mathscr{L}(u)$

$\mathscr{L}(\lambda^o x.t) = \lambda^o x.\mathscr{L}(t)$

Example

My First "Chris met Sandy" ACG Program

Σ_a :

NP, S :	*type*
CHRIS, SANDY :	NP
MET :	$NP \multimap NP \multimap S$

$NP := \sigma$

$S := \sigma$

CHRIS :=	*Chris*
SANDY :=	*Sandy*
MET :=	$\lambda^o os.s + met + o$

Σ_o :

σ :	*type*
$+$:	$\sigma \multimap \sigma \multimap \sigma$
Chris, *Sandy* :	σ
met :	σ

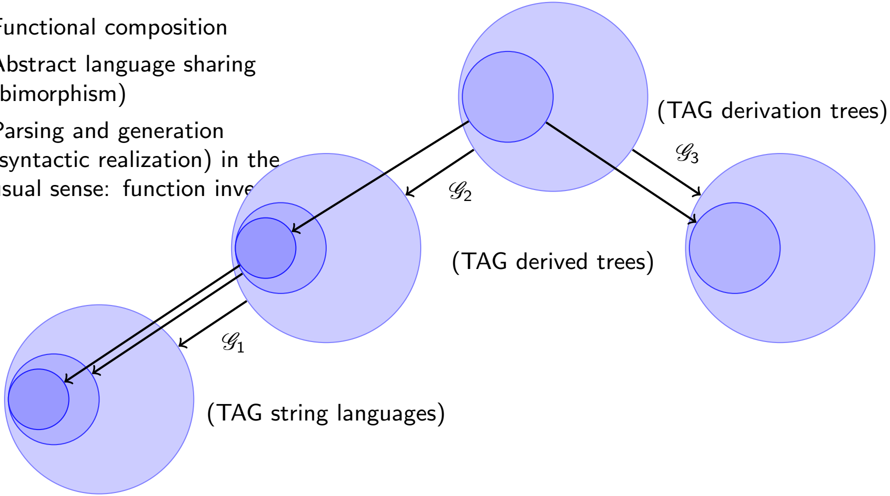
MET SANDY CHRIS : S

$$
\begin{aligned}
& \mathscr{L}(\text{MET SANDY CHRIS}) \\
&= \mathscr{L}(\text{MET})\, \mathscr{L}(\text{SANDY})\, \mathscr{L}(\text{CHRIS}) \\
&= (\lambda^o os.s + met + o)(Sandy)(Chris) \\
&= (\lambda^o s.s + met + Sandy)(Chris) \\
&= Chris + met + Sandy
\end{aligned}
$$

ACG Architecture

Composition Ability

- Functional composition
- Abstract language sharing (bimorphism)
- Parsing and generation (syntactic realization) in the usual sense: function inve



Intermediate Conclusion

So far...

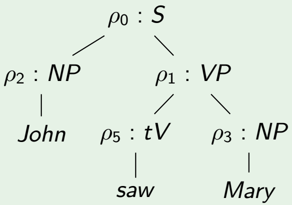
- Discussion on possible architectures of grammatical formalisms
- Discussion on function-compositional properties of ACG and *modularity*:
  - Abstract structures are mapped to object structures
  - Object structures: Strings, Simple semantic objects, Complex semantic objects:
    - Continuized semantics: $S := (t \multimap t) \multimap t$
    - Results of ACG composition
    - Dynamic semantics: $S := \gamma \multimap (\gamma \multimap t) \multimap t$ [de Groote(2006)]

      ARC CAuLD: Construction Automatique de représentations Logiques du Discours, `http://www.loria.fr/~pogodalla/cauld/`
    - Underspecified representations
  - Algorithms for parsing and generation (in the usual sense) are essentially the same: ACG parsing : finding the abstract antecedent of an object
  - Abstract structures?

CFG into ACG Encoding

Example (CFG)

$$
\begin{array}{lrcl}
\rho_0 : & S & \rightarrow & NP\ VP \\
\rho_1 : & VP & \rightarrow & tV\ NP \\
\rho_2 : & NP & \rightarrow & John \\
\rho_3 : & NP & \rightarrow & Mary \\
\rho_4 : & VP & \rightarrow & left \\
\rho_5 : & tV & \rightarrow & saw
\end{array}
$$



CFG as ACG

$$
\begin{array}{llcll}
 & \Sigma_{Rules} & \mathscr{L}_{\text{CFG}} & \Sigma_{Strings} & \\
\rho_0 & : NP \multimap VP \multimap S & := & \lambda xy.x + y & : \sigma \multimap \sigma \multimap \sigma \\
\rho_1 & : tV \multimap NP \multimap VP & := & \lambda xy.x + y & : \sigma \multimap \sigma \multimap \sigma \\
\rho_2 & : NP & := & John & : \sigma \\
\rho_3 & : NP & := & Mary & : \sigma \\
\rho_4 & : VP & := & left & : \sigma \\
\rho_5 & : tV & := & saw & : \sigma
\end{array}
$$

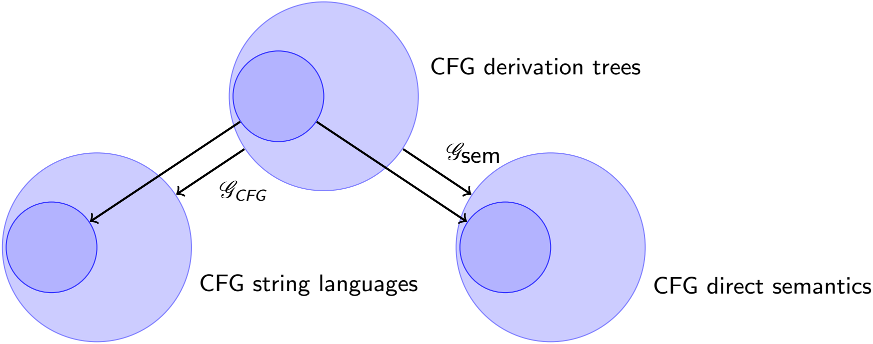
CFG into ACG Encoding (cont'd)

CFG as ACG

	Σ_{Rules}	$\mathscr{L}_{\mathrm{CFG}}$	$\Sigma_{Strings}$	
ρ_0	$: NP \multimap VP \multimap S$	$:=$	$\lambda xy.x + y$	$: \sigma \multimap \sigma \multimap \sigma$
ρ_1	$: tV \multimap NP \multimap VP$	$:=$	$\lambda xy.x + y$	$: \sigma \multimap \sigma \multimap \sigma$
ρ_2	$: NP$	$:=$	$John$	$: \sigma$
ρ_3	$: NP$	$:=$	$Mary$	$: \sigma$
ρ_4	$: VP$	$:=$	$left$	$: \sigma$
ρ_5	$: tV$	$:=$	saw	$: \sigma$

$$\begin{aligned}
\mathscr{L}_{\mathrm{CFG}}(\rho_0\ \rho_2\ (\rho_1\ \rho_5 \rho_3) : S) &= (\lambda xy.x + y) John((\lambda xy.x + y) saw\ Mary) \\
&\rightarrow_\beta (\lambda y.John + y)((\lambda y.saw + y) Mary) \\
&\rightarrow_\beta (\lambda y.John + y)(saw + Mary) \\
&\rightarrow_\beta John + (saw + Mary)
\end{aligned}$$

CFG Encoding



A Direct Semantics

Sharing Abstract Languages

CFG syntax as ACG

	Σ_{Rules}	$\mathscr{L}_{\mathrm{CFG}}$	$\Sigma_{Strings}$	
ρ_0	$: NP \multimap VP \multimap S$	$:=$	$\lambda xy.x + y$	$: \sigma \multimap \sigma \multimap \sigma$
ρ_1	$: tV \multimap NP \multimap VP$	$:=$	$\lambda xy.x + y$	$: \sigma \multimap \sigma \multimap \sigma$
ρ_2	$: NP$	$:=$	$John$	$: \sigma$
ρ_3	$: NP$	$:=$	$Mary$	$: \sigma$
ρ_4	$: VP$	$:=$	$left$	$: \sigma$
ρ_5	$: tV$	$:=$	saw	$: \sigma$

CFG (direct) semantics as ACG

	Σ_{Rules}	$\mathscr{L}_{\mathrm{sem}}$	Σ_{Log}	
ρ_0	$: NP \multimap VP \multimap S$	$:=$	$\lambda sP.P\, s$	$: e \multimap (e \multimap t) \multimap t$
ρ_1	$: tV \multimap NP \multimap VP$	$:=$	$\lambda Pos.P\, s\, o$	$: (e \multimap e \multimap t) \multimap e \multimap e \multimap t$
ρ_2	$: NP$	$:=$	**John**	$: e$
ρ_3	$: NP$	$:=$	**Mary**	$: e$
ρ_4	$: VP$	$:=$	**left**	$: e \multimap t$
ρ_5	$: tV$	$:=$	**saw**	$: e \multimap e \multimap t$

A Direct Semantics (cont'd)

CFG (direct) semantics as ACG

	Σ_{Rules}	$\mathscr{L}_{\text{sem}}$		Σ_{Log}
ρ_0	$: NP \multimap VP \multimap S$	$:=$	$\lambda sP.P\ s$	$: e \multimap (e \multimap t) \multimap t$
ρ_1	$: tV \multimap NP \multimap VP$	$:=$	$\lambda Pos.P\ s\ o$	$: (e \multimap e \multimap t) \multimap e \multimap e \multimap t$
ρ_2	$: NP$	$:=$	$\mathbf{John}$	$: e$
ρ_3	$: NP$	$:=$	$\mathbf{Mary}$	$: e$
ρ_4	$: VP$	$:=$	$\mathbf{left}$	$: e \multimap t$
ρ_5	$: tV$	$:=$	$\mathbf{saw}$	$: e \multimap e \multimap t$

$$
\begin{aligned}
\mathscr{L}_{\text{sem}}(\rho_0\ \rho_2\ (\rho_1\ \rho_5 \rho_3) : S) &= (\lambda sP.P\ s)\mathbf{John}((\lambda Pos.P\ s\ o)\mathbf{saw}\ \mathbf{Mary}) \\
&\to_\beta (\lambda P.P\ \mathbf{John})((\lambda os.\mathbf{saw}\ s\ o)\mathbf{Mary}) \\
&\to_\beta (\lambda P.P\ \mathbf{John})(\lambda^o s.\mathbf{saw}\ s\ \mathbf{Mary}) \\
&\to_\beta (\lambda^o s.\mathbf{saw}\ s\ Mary)\mathbf{John} \\
&\to_\beta \mathbf{saw}\ \mathbf{John}\ \mathbf{Mary}
\end{aligned}
$$

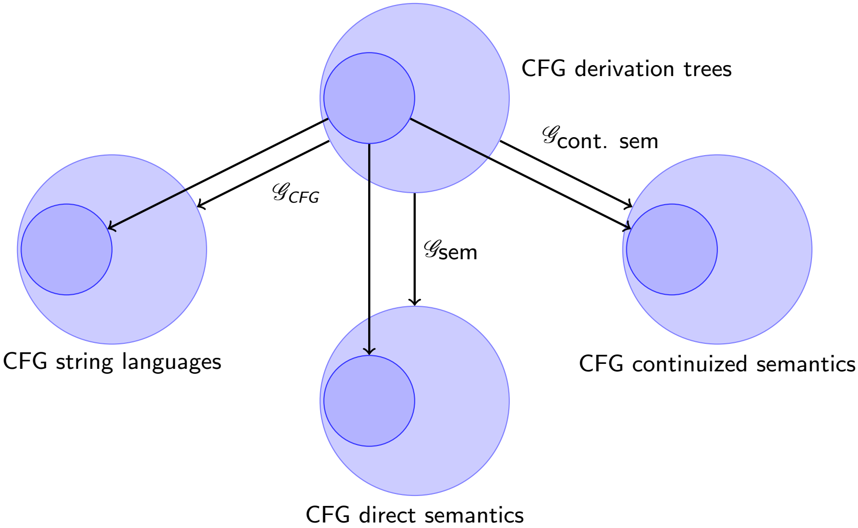
A Continuized (Higher–Order) Semantics

CFG continuized semantics as ACG [Barker(2002)]

	Σ_{Rules}	$\mathscr{L}_{sem}$	Σ_{Log}
S		$:=$	$(t \multimap t) \multimap t$
NP		$:=$	$(e \multimap t) \multimap t$
VP		$:=$	$((e \multimap t) \multimap t) \multimap t$
tV		$:=$	$((e \multimap e \multimap t) \multimap t) \multimap t$
N		$:=$	$((e \multimap t) \multimap t) \multimap t$
Det		$:=$	$(((e \multimap t) \multimap t) \multimap t) \multimap (e \multimap t) \multimap t$
ρ_0	$: NP \multimap VP \multimap S$	$:=$	$\lambda^o svp.v(\lambda^o P.s(\lambda^o x.p(Px)))$
ρ_1	$: tV \multimap NP \multimap VP$	$:=$	$\lambda^o voP.v(\lambda^o R.o(\lambda^o y.P(Ry)))$
ρ_2	$: NP$	$:=$	$\lambda^o P.P\ \mathbf{John}$
ρ_5	$: tV$	$:=$	$\lambda^o P.P\ \mathbf{saw}$
ρ_{every}	$: Det$	$:=$	$\lambda^o KP.K(\lambda^o Q.\forall x.(Q\,x) \Rightarrow (P\,x))$

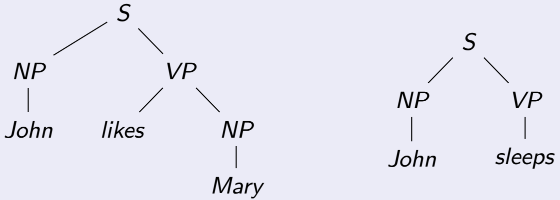
- Scope ambiguity
- Scope displacement
- NP as a scope island

CFG Encoding



Trees as λ-Terms

Trees Build on a Ranked Alphabet



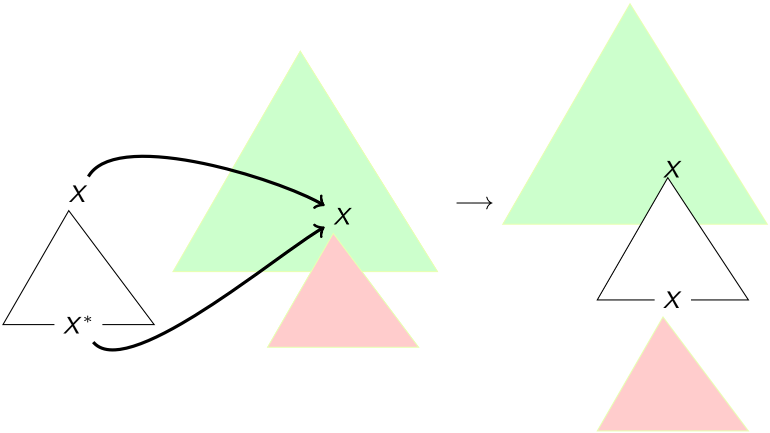
$S_2(NP_1\,John)(VP_2\,likes\,(NP_1\,Mary))$ $\quad$ $S_2(NP_1\,John)(VP_1\,sleeps)$

- S of arity 2 (non-terminal)
- NP of arity 1 (non-terminal)
- VP? VP_1 of arity 1 and VP_2 of arity 2 (non-terminals)
- *John* of arity 0 (terminal)

- $S_2 : \tau \multimap \tau \multimap \tau$
- $NP_1 : \tau \multimap \tau$
- $VP_1 : \tau \multimap \tau$, $VP_2 : \tau \multimap \tau \multimap \tau$
- $John : \tau$

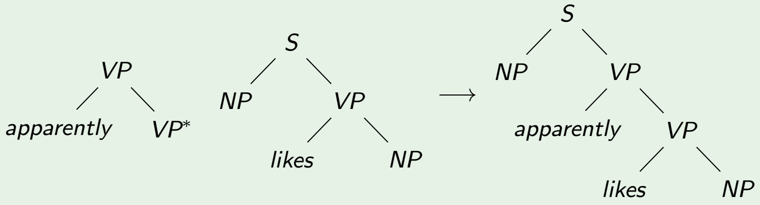
A Functional View on TAG

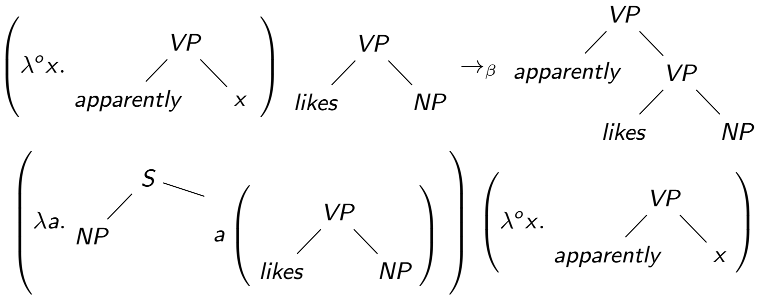
Tree Adjunction:



Auxiliary Trees as Functions

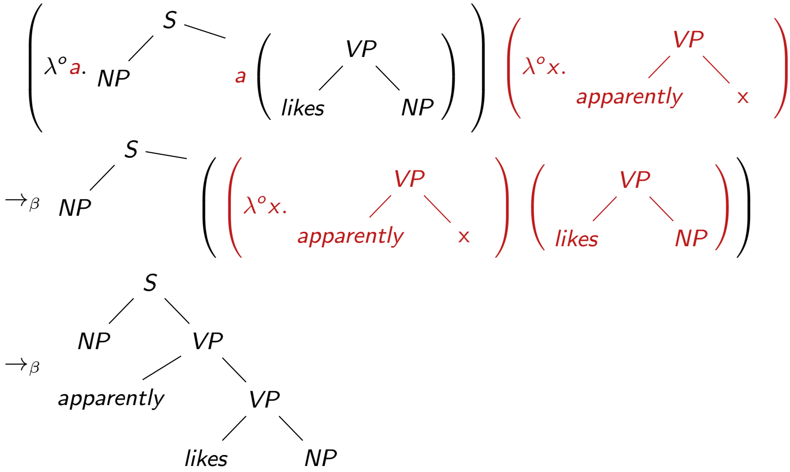
Example



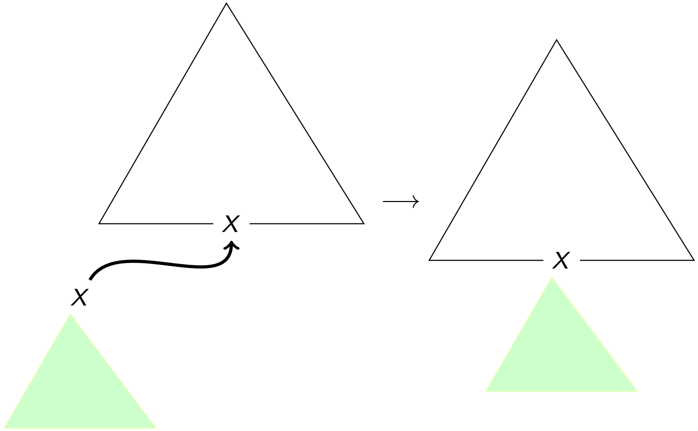


Adjunction as Functional Application

$\gamma'_{likes}\ \gamma'_{apparently} =$

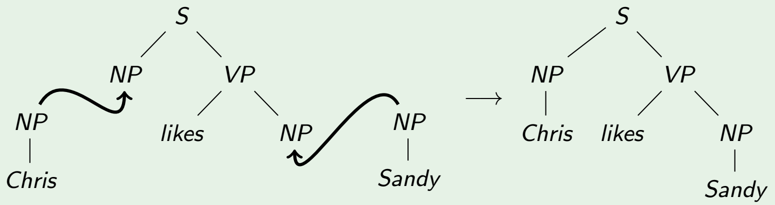


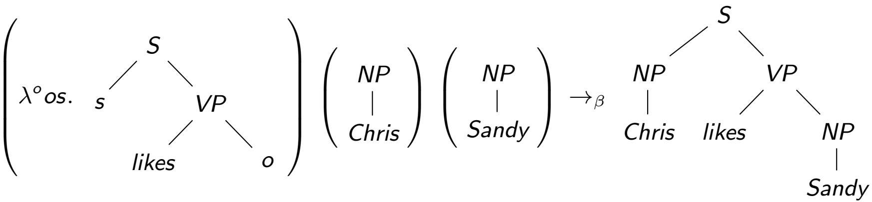
Substitution Operation



Substitution as Functional Application

Example





Putting Everything Together

Σ_{trees}:

$\tau \quad : type$

$\gamma_{apparently} = \lambda^o ax.a\left(\begin{array}{c} VP \\ apparently \quad x \end{array}\right) \quad : (\tau \multimap \tau) \multimap \tau \multimap \tau$

$I = \lambda^o x.x \quad : \tau \multimap \tau$

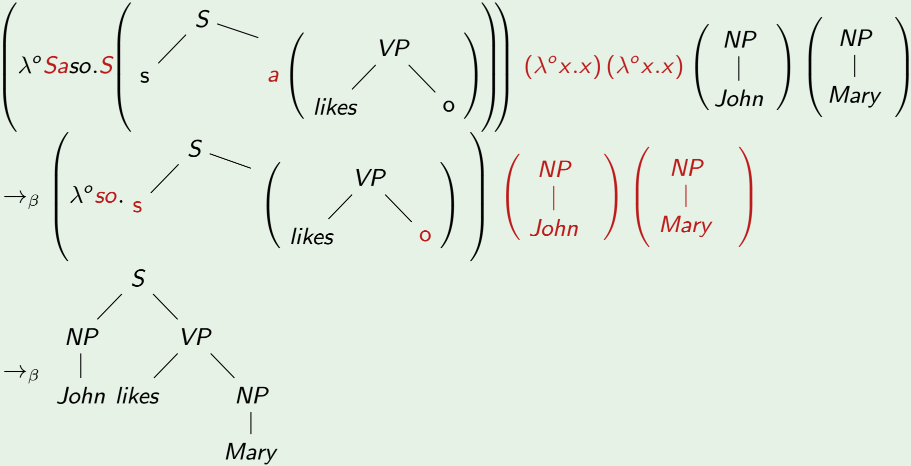
$\gamma_{John} = \begin{array}{c} NP \\ | \\ John \end{array} \quad : \tau$

$\gamma_{likes} = \lambda^o Saso.S\left(\begin{array}{c} S \\ s \quad a\left(\begin{array}{c} VP \\ likes \quad o \end{array}\right) \end{array}\right) \quad : (\tau \multimap \tau) \multimap (\tau \multimap \tau) \multimap \tau \multimap \tau \multimap \tau$

TAG Derivation as Term Application

Example

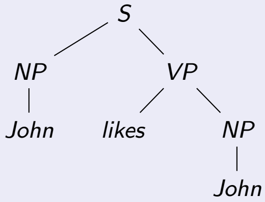
$\gamma_{likes}\ I\ I\ \gamma_{John}\ \gamma_{Mary} =$



Yield as an ACG

Derived Tree Signature

- $S_2 : \tau \multimap \tau \multimap \tau$
- $NP_1 : \tau \multimap \tau$
- $VP_1 : \tau \multimap \tau,\ VP_2 : \tau \multimap \tau \multimap \tau$
- $John : \tau$



$S_2 (NP_1\, John)(VP_2\, likes\, (NP_1\, Mary))$

String signature (as before):

σ	: type
John, likes ...	: σ

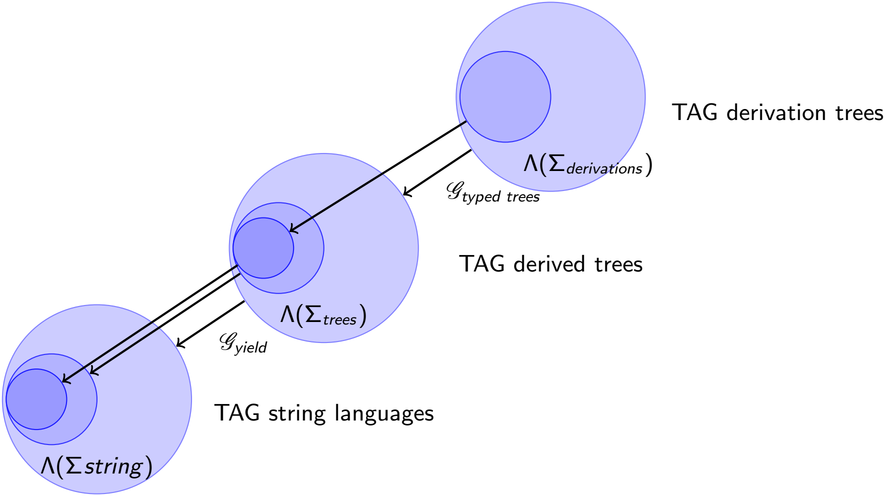
$\mathscr{G}_{Yield}$

τ	$:= \sigma$	$John$	$:= John$
X_1	$:= \lambda x.x$	X_2	$:= \lambda xy.x + y$
...			

$$S_2 (NP_1\, John)(VP_2\, likes\, (NP_1\, Mary)) := John + (likes + Mary)$$

TAG as ACG

The Current Picture



$\mathcal{A}(\mathscr{G}_{yield}) =$ TAG Derived Trees?

Σ_{trees}:

$$\tau \quad : type$$

$$\gamma_{apparently} = \lambda^o ax.a\left(\begin{array}{ccc} & VP & \\ \swarrow & & \searrow \\ apparently & & x \end{array}\right) : (\tau \multimap \tau) \multimap \tau \multimap \tau$$

$$I = \lambda^o x.x \quad : \tau \multimap \tau$$

$$\gamma_{John} = \begin{array}{c} NP \\ | \\ John \end{array} \quad : \tau$$

$$\gamma_{apparently}\, I\, \gamma_{John} = \begin{array}{ccc} & VP & \\ \swarrow & & \searrow \\ apparently & & NP \\ & & | \\ & & John \end{array}$$

TAG as ACG

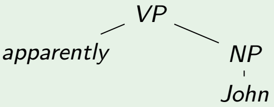
Category Induced Constraints

- The site of an adjunction has the same category as the root (and foot) node of the auxiliary tree
- The site of a substitution has the same category as the root node of the substituted tree

	$\Sigma_{derivations}$	$\xrightarrow{\mathscr{L}_{\text{typed trees}}}$		Σ_{trees}
c_{John}	$: NP$	$:=$	γ_{John}	$: \tau$
$c_{apparently}$	$: (VP \multimap VP) \multimap VP \multimap VP$	$:=$	$\gamma_{apparently}$	$: (\tau \multimap \tau) \multimap \tau \multimap \tau$
$NP, VP, S \ldots$	: types	$:=$	σ	

Example

There is no $t : VP \in \Lambda(\Sigma_{derivations})$ such that $t :=$

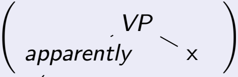


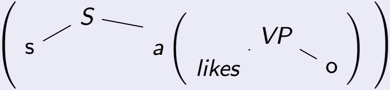
Control on the Derived Trees

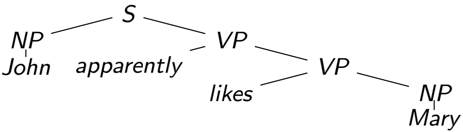
$\mathscr{G}_{\textit{typed trees}} = \langle \Sigma_{\textit{derivations}}, \Sigma_{\textit{trees}}, \mathscr{L}_{\text{typed trees}}, S \rangle$

$NP, VP, S \ldots \quad := \sigma$

$c_{John} : NP \quad :=$ 

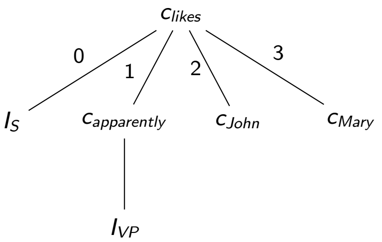
$c_{apparently} : (VP \multimap VP) \multimap VP \multimap VP \quad := \lambda^o ax.a$ 

$c_{likes} : (S \multimap S) \multimap (VP \multimap VP) \multimap NP \multimap NP \multimap S \quad := \lambda^o Saso.S$ 

$c_{likes}(c_{apparently}\, I_{VP})\, c_{John}\, c_{Mary} : S =$ 

TAG Derivation Trees as Abstract Terms

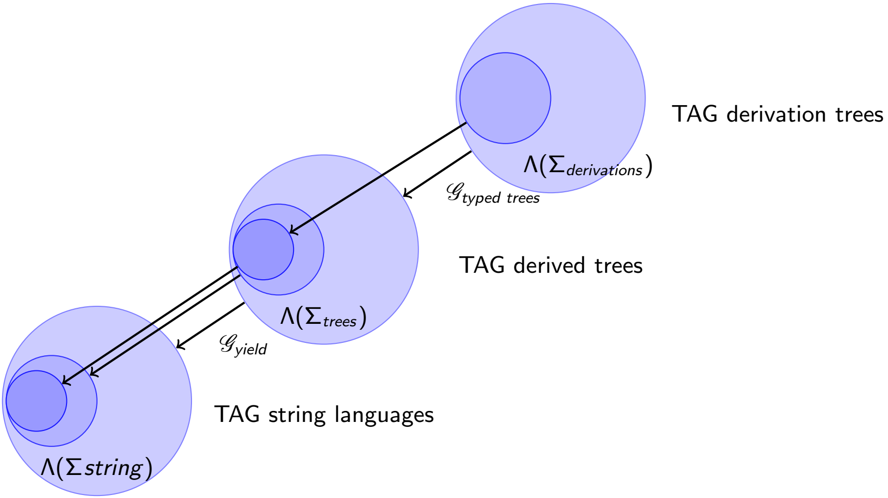
$c_{likes}\ I_S(c_{apparently}\ I_{VP})\ c_{John}\ c_{Mary} : S \quad =$



$$c_{likes} \quad : \underbrace{(S \multimap S)}_{0} \multimap \underbrace{(VP \multimap VP)}_{1} \multimap \underbrace{NP}_{2} \multimap \underbrace{NP}_{3} \multimap S$$

TAG as ACG

Intermediate Picture



Let's Build Some Semantic Representation

Forgetting few seconds about TAG, we have:

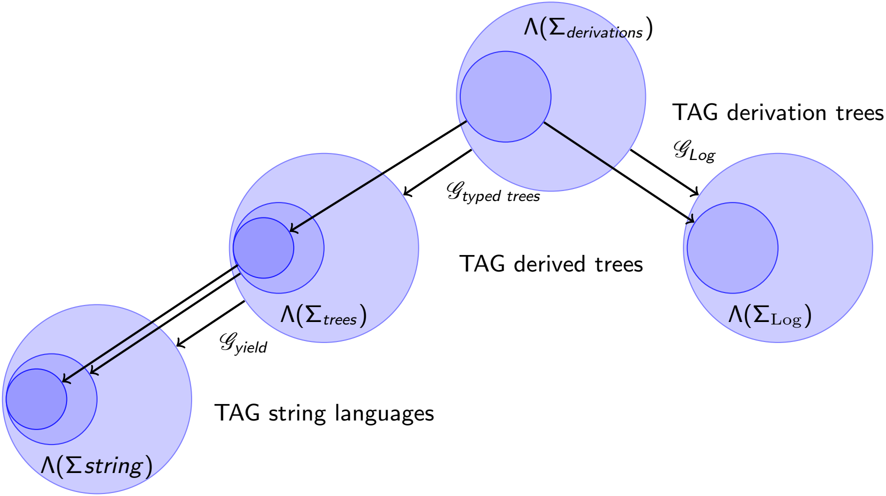
- A higher-order signature $\Sigma_{derivations}$:
$$\begin{array}{ll} VP, NP, S & : \text{types} \\ c_{john}, c_{mary} & : NP \\ c_{apparently} & : VP \multimap VP \\ c_{likes} & : (VP \multimap VP) \multimap NP \multimap S \end{array}$$
- Some knowledge about Montague-like semantics?

A standard interpretation

$$\begin{array}{llll} S & := t & NP & := (e \multimap t) \multimap t \\ VP & := e \multimap t & & \\ c_{john} & := \lambda^o P.P\,\mathbf{j} & c_{apparently} & := \lambda^o aP.a(\lambda x.\mathbf{apparently}(P\,x)) \\ I_{VP} & := \lambda x.x & c_{likes} & := \lambda aos.s(a(\lambda x.o(\lambda y.\mathbf{like}\,x\,y))) \end{array}$$

How to get the object wide scope reading?

TAG with Semantics



Intermediate Conclusion

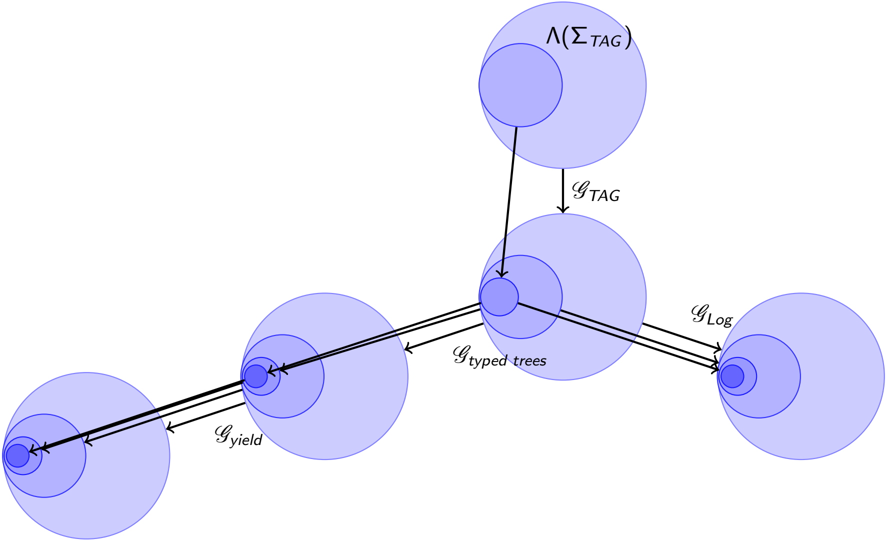
So far

- Trees as λ-terms
- Yield as an ACG
- Typing control: ACG from derivation trees to derived trees
- Some semantics added. Is it a function from syntax?

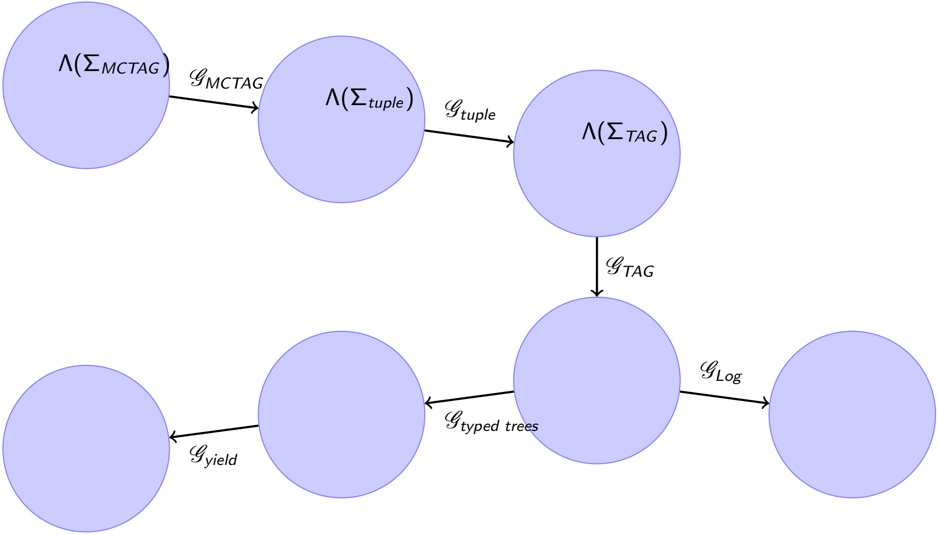
Questions?

- Any TAG feature missing?
- Order of $\mathscr{G}_{typed\ trees}$? ($c_{likes} : (VP \multimap VP) \multimap NP \multimap S$)

The Actual Picture



The Final Picture



Scope Ambiguities

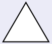
every man loves some woman $\rightarrow \left\{ \begin{array}{l} \forall x.\textbf{man}\,x \rightarrow (\exists y.\textbf{woman}\,y \wedge \textbf{love}\,x\,y) \\ \exists y.\textbf{woman}\,y \wedge (\forall x.\textbf{man}\,x \rightarrow \textbf{love}\,x\,y) \end{array} \right.$

CFG-like systems

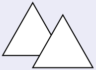
$$[\![loves]\!] = \lambda os.s(\lambda x.o(\lambda y.\textsc{love}(x,y)))$$

$$[\![loves]\!] = \lambda os.o(\lambda y.s(\lambda x.\textsc{love}(x,y)))$$

Underspecified framework (see the section on underspecification)

every man loves some woman $\rightarrow$  $\rightarrow$? $\rightarrow \left\{ \begin{array}{l} \forall x.\textbf{man}\,x \rightarrow (\exists y.\textbf{woman}\,y \wedge \textbf{love}\,x\,y) \\ \exists y.\textbf{woman}\,y \wedge (\forall x.\textbf{man}\,x \rightarrow \textbf{love}\,x\,y) \end{array} \right.$

Type Logical framework

every man loves some woman 

$\rightarrow \forall x.\textbf{man}\,x \rightarrow (\exists y.\textbf{woman}\,y \wedge \textbf{love}\,x\,y)$

$\rightarrow \exists y.\textbf{woman}\,y \wedge (\forall x.\textbf{man}\,x \rightarrow \textbf{love}\,x\,y)$

Strengths and Weaknesses

Underspecified framework

Pros:

- One syntactic analysis
- Expressivity

Cons:

- Description language
- Ambiguity in the semantic recipe, not in the interface

TL framework

Pros:

- No intermediate language
- Ambiguity handled by the process

Cons:

- Syntactic ambiguity

Question

Is there an ACG way providing a proof-theoretic approach with only one syntactic structure and no intermediate language?

Scope Ambiguity in Categorial grammars

The standard way

loves
$NP\backslash S/NP \quad [NP]$
one $\quad NP\backslash S$
$P\backslash S)$
S
S/NP $\qquad$ *someone* $(NP/S)\backslash S$
S
$C_{someone}\,(\lambda^o y.C_{everyone}\,(\lambda^o x.C_{loves}\,y\,x))$

loves
$[NP] \quad NP\backslash S/NP$ $\qquad$ *som*
$S/NP \qquad (NP$
everyone $\quad S$
$S/(NP\backslash S) \quad NP\backslash S$
S
$C_{everyone}\,(\lambda^o x.C_{someone}\,(\lambda^o y.C_{loves}\,y\,x))$

The ACG way

- Replace $\backslash$ and $/$ by $\multimap$
- $C_{everyone} : (NP \multimap S) \multimap S$

Scope Ambiguity in ACG: The Semantics

Σ_{CG}

$$
\begin{array}{ll}
C_{loves} & : NP \multimap NP \multimap S \\
C_{everyone} & : (NP \multimap S) \multimap S \\
C_{someone} & : (NP \multimap S) \multimap S
\end{array}
$$

$\longrightarrow$

Σ_{log}

$$
\begin{array}{ll}
\textsc{loves} & : e \multimap e \multimap t \\
\forall, \exists & : (e \rightarrow t) \multimap \\
\wedge, \Rightarrow & : t \multimap t \multimap t
\end{array}
$$

$\mathscr{L}_{\text{amb-log}}$

$$
\begin{array}{ll}
NP & := e \\
S & := t \\
C_{loves} & := \lambda^o os.\textsc{loves}(s.o) \\
C_{everyone} & := \lambda^o P.\forall x.\textsc{human}(x) \Rightarrow P\ x \\
C_{someone} & := \lambda^o P.\exists y.\textsc{human}(y) \wedge P\ y
\end{array}
$$

$$
\begin{array}{l}
C_{everyone}(\lambda^o x.C_{someone}(\lambda^o y.C_{loves}\ x\ y)) \\
\quad :=_{\text{amb-log}} (\lambda^o P.\forall x.\textsc{human}(x) \Rightarrow P\ x)((\lambda^o x.(\lambda^o P.\exists y.\textsc{human}(y) \wedge P\ y)(\lambda^o y.\textsc{loves}(x,y)))) \\
\quad \rightarrow_\beta (\lambda^o P.\forall x.\textsc{human}(x) \Rightarrow P\ x)((\lambda^o x.(\exists y.\textsc{human}(y) \wedge \textsc{loves}(x,y)))) \\
\quad \rightarrow_\beta (\forall x.\textsc{human}(x) \Rightarrow (\exists y.\textsc{human}(y) \wedge \textsc{loves}(x,y)))
\end{array}
$$

Scope Ambiguity in ACG: The Semantics (cont'd)

Σ_{CG}

$$\begin{array}{ll} C_{loves} & : NP \multimap NP \multimap S \\ C_{everyone} & : (NP \multimap S) \multimap S \\ C_{someone} & : (NP \multimap S) \multimap S \end{array}$$

$\longrightarrow$ $\mathscr{L}_{\text{amb-log}}$

Σ_{log}

$$\begin{array}{ll} \text{LOVES} & : e \multimap e \multimap t \\ \forall, \exists & : (e \rightarrow t) \multimap \\ \wedge, \Rightarrow & : t \multimap t \multimap t \end{array}$$

$\mathscr{L}_{\text{amb-log}}$

$$\begin{array}{ll} NP & := e \\ S & := t \\ C_{loves} & := \lambda^o os.\text{LOVES}(s.o) \\ C_{everyone} & := \lambda^o P.\forall x.\text{HUMAN}(x) \Rightarrow P\ x \\ C_{someone} & := \lambda^o P.\exists y.\text{HUMAN}(y) \wedge P\ y \end{array}$$

$$\begin{array}{l} C_{someone}(\lambda^o y.C_{everyone}(\lambda^o x.C_{loves}\, x\, y)) \\ \quad :=_{\text{amb-log}} (\lambda^o P.\exists y.\text{HUMAN}(y) \wedge P\ y)((\lambda^o x.(\lambda^o P.\forall x.\text{HUMAN}(x) \Rightarrow P\ y)(\lambda^o y.\text{LOVES}(x,y)))) \\ \quad \rightarrow_\beta (\lambda^o P.\exists y.\text{HUMAN}(y) \wedge P\ x)((\lambda^o x.(\forall x.\text{HUMAN}(x) \Rightarrow \text{LOVES}(x,y)))) \\ \quad \rightarrow_\beta (\exists y.\text{HUMAN}(y) \wedge (\forall x.\text{HUMAN}(x) \Rightarrow \text{LOVES}(x,y))) \end{array}$$

Scope Ambiguity in ACG

Σ_{CG}

$$\begin{array}{ll} C_{loves} & : NP \multimap NP \multimap S \\ C_{everyone} & : (NP \multimap S) \multimap S \\ C_{someone} & : (NP \multimap S) \multimap S \end{array}$$

$\longrightarrow$

Σ_{string}

$$\begin{array}{ll} loves & : \sigma \\ everyone & : \sigma \\ someone & : \sigma \end{array}$$

$\mathscr{L}_{\text{amb-string}}$

$$\begin{array}{ll} C_{loves} & := \lambda^o os.s + loves + o \\ C_{everyone} & := \lambda^o P.P\ everyone \\ C_{someone} & := \lambda^o P.P\ someone \end{array}$$

$$\begin{array}{l} C_{everyone}(\lambda^o x.C_{someone}(\lambda^o y.C_{loves}\ x\ y)) \\ \quad :=_{\text{amb-string}} (\lambda^o P.P\ everyone)((\lambda^o x.\lambda^o P.P\ someone)(\lambda^o y.(\lambda^o os.s + loves + o)\ y\ x)) \\ \quad \rightarrow_\beta (\lambda^o P.P\ everyone)((\lambda^o x.\lambda^o P.P\ someone)(\lambda^o y.x + loves + y)) \\ \quad \rightarrow_\beta (\lambda^o P.P\ everyone)(\lambda^o x.(\lambda^o y.x + loves + y)\ someone) \\ \quad \rightarrow_\beta (\lambda^o P.P\ everyone)(\lambda^o x.x + loves + someone) \\ \quad \rightarrow_\beta (\lambda^o x.x + loves + someone)\ everyone \\ \quad \rightarrow_\beta everyone + loves + someone \end{array}$$

Scope Ambiguity in ACG (cont'd)

Σ_{CG}

$C_{loves} : NP \multimap NP \multimap S$

$C_{everyone} : (NP \multimap S) \multimap S$

$C_{someone} : (NP \multimap S) \multimap S$

$\longrightarrow$

$\mathscr{L}_{\text{amb-string}}$

$C_{loves} := \lambda^o os.s + loves + o$

$C_{everyone} := \lambda^o P.P\ everyone$

$C_{someone} := \lambda^o P.P\ someone$

Σ_{string}

$loves : \sigma$

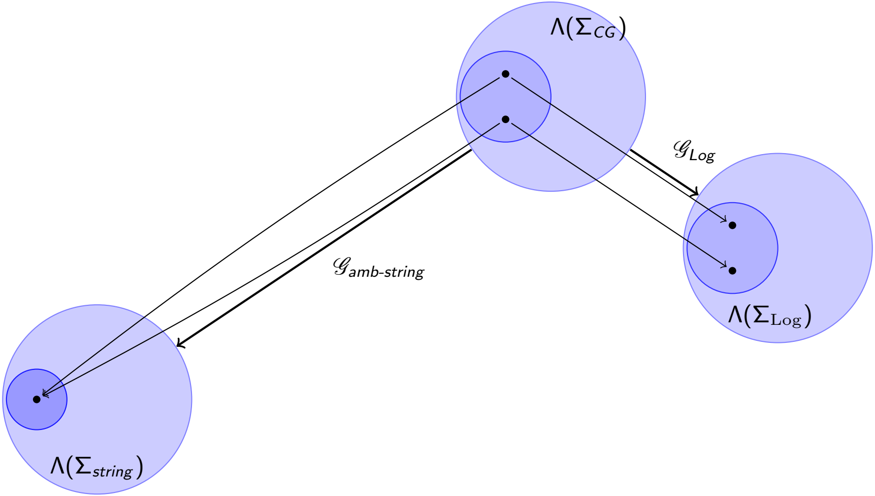
$everyone : \sigma$

$someone : \sigma$

$$
\begin{aligned}
& C_{someone}(\lambda^o y.C_{someone}(\lambda^o x.C_{loves}\ x\ y)) \\
& :=_{\text{amb-string}} (\lambda^o P.P\ someone)((\lambda^o y.\lambda^o P.P\ everyone)(\lambda^o x.(\lambda^o os.s + loves + o)\ y\ x)) \\
& \rightarrow_\beta (\lambda^o P.P\ someone)((\lambda^o y.\lambda^o P.P\ everyone)(\lambda^o x.x + loves + y)) \\
& \rightarrow_\beta (\lambda^o P.P\ someone)(\lambda^o y.(\lambda^o x.x + loves + y)\ everyone) \\
& \rightarrow_\beta (\lambda^o P.P\ someone)(\lambda^o y.everyone + loves + y) \\
& \rightarrow_\beta (\lambda^o y.everyone + loves + x)\ someone \\
& \rightarrow_\beta everyone + loves + someone
\end{aligned}
$$

Scope Ambiguity

Non Injective Lexicon



Scope Ambiguity in ACG (cont'd)

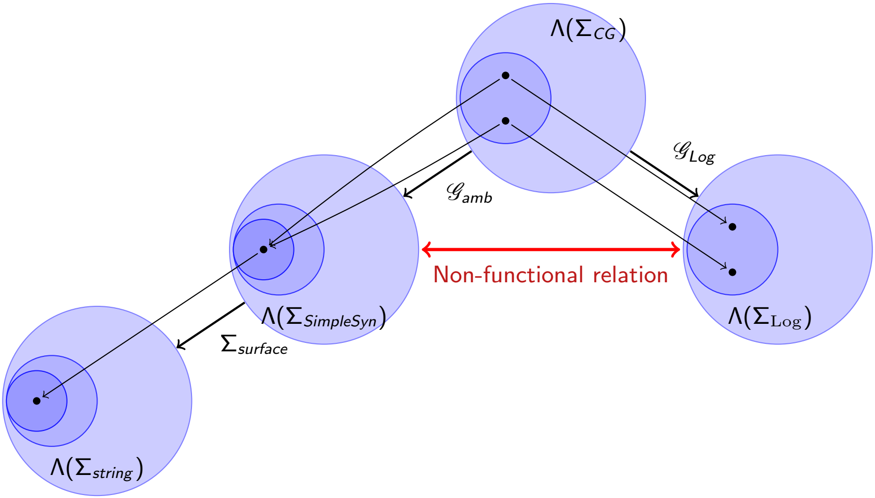
$$
\begin{array}{ll}
\Sigma_{CG} & \\
\quad C_{loves} & : NP \multimap NP \multimap S \\
\quad C_{everyone} & : (NP \multimap S) \multimap S \\
\quad C_{someone} & : (NP \multimap S) \multimap S
\end{array}
\longrightarrow
\begin{array}{ll}
\Sigma_{SimpleSyn} & \\
\quad c_{loves} & : NP \multimap NP \multimap S \\
\quad c_{everyone} & : NP \\
\quad c_{someone} & : NP
\end{array}
$$

$$
\begin{array}{ll}
\mathscr{L}_{\mathrm{amb}} & \\
\quad C_{loves} & := \lambda^o os.loves\ o\ s \lambda^o os.c_{loves}\ o\ s \\
\quad C_{everyone} & := \lambda^o P.P\ everyone c_{everyone} \\
\quad C_{someone} & := \lambda^o P.P\ someone c_{someone}
\end{array}
$$

$$
\begin{array}{ll}
C_{everyone}(\lambda^o x.C_{someone}(\lambda^o y.C_{loves}\ x\ y)) & :=_{\mathrm{amb}} c_{loves}\ c_{someone}\ c_{everyone} \\
C_{someone}(\lambda^o y.C_{someone}(\lambda^o x.C_{loves}\ x\ y)) & :=_{\mathrm{amb}} c_{loves}\ c_{someone}\ c_{everyone}
\end{array}
$$

Scope Ambiguity

Non Injective Lexicon



Conjunction

John and every kid ran

$$
\begin{array}{ll}
C_{and} & := \lambda^o PQR.P(\lambda^o x.Q(\lambda^o y.R(c_{and}\, x\, y))) \\
C_{run} & := c_{run} \\
C_{kid} & := c_{kid} \\
C_{John} & := c_{John}
\end{array}
$$

$$
\begin{array}{ll}
C_{run} & : NP \multimap S \\
C_{kid} & : N \\
C_{John} & : NP \\
C_{and} & : ((NP \multimap S) \multimap S) \\
 & \multimap ((NP \multimap S) \multimap S) \\
 & \multimap (NP \multimap S) \multimap S
\end{array}
$$

$$C_{and}(\lambda^o P.P C_{John})(c_{every}\, c_{kid})\, C_{run}$$

$$\longleftarrow$$

$$
\begin{array}{ll}
c_{run} & : NP \multimap S \\
c_{kid} & : N \\
c_{John} & : NP \\
c_{and} & : NP \multimap NP \multimap NP
\end{array}
$$

$$c_{run}(c_{and}\, c_{John}(c_{every}\, c_{kid}))$$

$$
\begin{array}{ll}
c_{run} & := \lambda^o s.s + ran \\
c_{kid} & := kid \\
c_{John} & := John \\
c_{and} & := \lambda^o xy.x + and + y
\end{array}
$$

$$
\begin{array}{ll}
C_{run} & := \mathbf{run} \\
C_{kid} & := \mathbf{kid} \\
C_{John} & := \mathbf{j} \\
C_{and} & := \lambda^o PQ. \\
 & \quad \lambda R.(PR) \wedge (QR)
\end{array}
$$

$$\Sigma_{string}$$

$$(John + and + (every + kid)) + ran$$

$$\Sigma_{Log}$$

$$(\mathbf{run}\,\mathbf{j}) \wedge (\forall x.\mathbf{kid}\, x \Rightarrow \mathbf{run}\, x)$$

De re and *De dicto* Readings

$$\begin{array}{ll} C_{seek} & := \lambda^o xP.P(\lambda^o y.c_{seek}\, x\, y) \\ C_{book} & := c_{book} \end{array}$$

$$\begin{array}{ll} C_{seek} & : NP \multimap ((NP \multimap S) \multimap S) \multimap S \\ C_{book} & : N \end{array}$$

$C_{seek}\, C_{John}\, (C_a\, C_{book})$

$(C_a\, C_{book})(\lambda^o y.C_{seek}\, C_{John}\, (\lambda^o Q.Q\, y))$

$$\begin{array}{ll} c_{seek} & : NP \multimap NP \multimap S \\ c_{book} & : N \end{array}$$

$c_{seek}\, c_{John}\, (c_a\, c_{book})$

$$\begin{array}{ll} c_{seek} & := \lambda^o xy.x + \mathit{seeks} + y \\ c_{book} & := \mathit{book} \end{array}$$

$$\begin{array}{ll} C_{seek} & := \lambda^o xo.\, \mathbf{try}\, x\, (\lambda^o z.o\, (\lambda^o y.\mathbf{find}\, z\, y)) \\ C_{book} & := \mathbf{book} \end{array}$$

Σ_{string}

$\mathit{John} + \mathit{seeks} + (a + \mathit{book})$

Σ_{Log}

$\exists y.(\mathbf{book}\, y) \wedge (\mathbf{try}\, \mathbf{j}\, (\lambda^o x.\mathbf{find}\, x\, y))$

$\mathbf{try}\, \mathbf{j}\, (\lambda^o x.\exists y.(\mathbf{book}\, y) \wedge (\mathbf{find}\, x\, y))$

And More...

So far

- Conjunction
- *De re* and *de dicto* readings
- Coordination of quantified and non-quantified NPs
- VP ellipsis *John saw a kid and so did Bill*
- *de re* and *de dicto* readings
- Quantification and negation (*every kid did'nt run*)

Generalization: the Scoping Constructor

$$\frac{\Gamma \vdash_{\mathrm{TL}} t : \beta \Uparrow \alpha \qquad \Delta, x : \beta \vdash_{\mathrm{TL}} u : \alpha}{\Gamma, \Delta \vdash_{\mathrm{TL}} t(\lambda x.u) : \alpha} (\mathrm{E}_{\Uparrow}) \quad \frac{\Gamma \vdash_{\mathrm{TL}} t : \beta}{\Gamma \vdash_{\mathrm{TL}} \lambda x.(x\, t) : \beta \Uparrow \alpha} (\mathrm{I}_{\Uparrow})$$

Syntactically behaves as a β and semantically as a $(\beta \multimap \alpha) \multimap \alpha$

Given a lexical entre $w : a \in TL(A)$, we have $c_w : a^{\mathrm{syn}}$ and $C_w : a^{\mathrm{sem}}$ such that:

- if $a \in A$ then $a^{\mathrm{syn}} = a$ and $a^{\mathrm{sem}} = a$
- if $a = \alpha \multimap \beta$ then $a^{\mathrm{syn}} = \alpha^{\mathrm{syn}} \multimap \beta^{\mathrm{syn}}$ and $a^{\mathrm{sem}} = \alpha^{\mathrm{sem}} \multimap \beta^{\mathrm{sem}}$.
- if $a = \alpha \Uparrow \beta$ then $a^{\mathrm{syn}} = \alpha^{\mathrm{syn}}$ and $a^{\mathrm{sem}} = (\alpha^{\mathrm{sem}} \multimap \beta^{\mathrm{sem}}) \multimap \beta^{\mathrm{sem}}$

Conclusion on ACG

- A large number of grammatical formalisms can be encoded into (2nd order) ACG
- Semantic ambiguity in type-logical grammars arises from higher-order (3rd order) types
- Type-logical grammars can be provided a "simple syntactic" level of 2nd order
- We can apply the same higher-order technics to any 2nd order ACG!
- Encoding in a same framework: sharing and comparing analysis
- ACG composition modes: flexible and open architectures
- "Syntax", "function", "relation", "compositionality", "rule-to-rule" intuitions may be realized by different mathematical notions
- Shallow opposition between *syntactocentric* and *parallel* formalisms

Main Features [Egg(2010)]

Principles

- **Principle** To omit some information from linguistic descriptions
- **Aim** To capture alternative realisations in one single representation
- **Aim** To avoid enumeration of the alternatives

Example

- Morphological features in grammar rules:

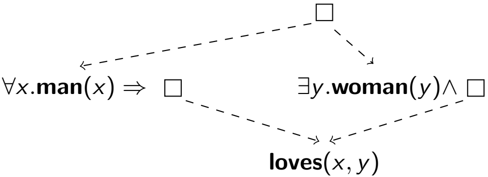
$$S \longrightarrow NP_{\text{nom}}\ VP$$
$$NP_{\text{nom}} \longrightarrow John$$
$$NP_{\text{acc}} \longrightarrow John$$

- Unification grammars:

$$S \longrightarrow NP[\textbf{case} = \text{nom}]\ VP$$
$$NP \longrightarrow John$$

Features [Bos(1995)]

- Two levels of description:
  - an object-level of linguistic representations
  - a meta-level of describing these representations
- Fragments of object-level semantic representations
- Glue points (hole) in the fragments
- Relation between glue points and fragments



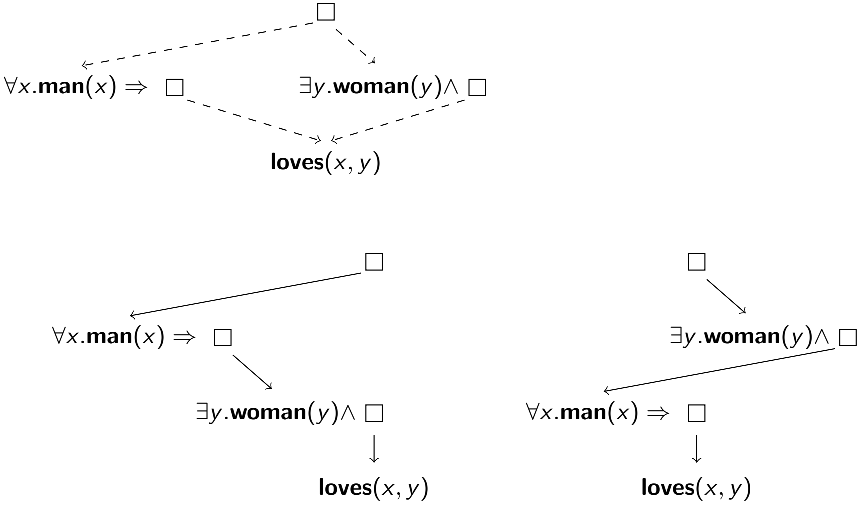
Description and Models

Getting the Readings

- Each hole gets filled by a fragment
- Respecting the description

We get a model a description

Semantic Ambiguity: Example



Accessibility

Anaphoric pronouns and their antecedents

Example (Existentials, proper nouns, and negation)

- John owns a car. It is red.
- $\exists x\, \mathbf{car}\, x \wedge \mathbf{own}\, \mathbf{j}\, x \wedge \mathbf{red}\, x$
- John doesn't own a car. *It is red.
- $\neg(\exists x\, \mathbf{car}\, x \wedge \mathbf{own}\, \mathbf{j}\, x) \wedge \mathbf{red}\, x$
- John doesn't own a car. He is ecology-minded.
- $\neg(\exists x\, \mathbf{car}\, x \wedge \mathbf{own}\, \mathbf{j}\, x) \wedge \mathbf{ecolo}\, \mathbf{j}$

What we've learned from DRT:

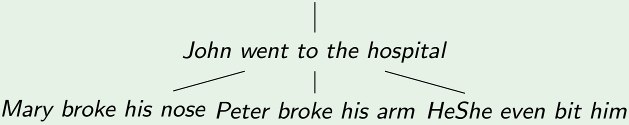
- Indefinite noun phrases (existentials) introduce discourse referents
- Negation limits the accessibility of discourse referents (existentials $\neq$ proper nouns)

Accessibility

Anaphoric Pronouns and Their Antecedents

Example (Hierarchical structure of the discourse [Busquets et al.(2001)Busquets, Vieu, and Asher])

- John went to the hospital.
- Mary broke his nose.
- Peter broke his arm
- He*She even bit him.



What we've learned from theories on rhetorical structure

- Segments of the discourse stand in relation to each other
- Depending on the relation (*coordinating*, *subordinating*), discourse markers are accessible or not

Dynamic Logics

Technical Issues

- Non-standard interpretation of formulas:
  - Interpretation as relations between assignment functions
  - $[\![\phi \rightarrow \psi]\!] = \{\langle g, h\rangle | h = g \& \forall k.h[\![\phi]\!]k \Rightarrow \exists j.k[\![\psi]\!]j\}$
  - $(\exists x.\phi) \wedge \psi \Leftrightarrow \exists x.(\phi \wedge \psi)$ (scope theorem)
- Destructive assignment and variable clash
- $[\![\phi \Rightarrow_{\diamond} \psi]\!] =?$

Formal Semanticist or Logician?

- What are the useful data to feed the context with?
- How do discourse and sentences combine?
- What are the semantic recipes of the lexical items
- Should I design a new logic? Continuation semantics

Continuation Semantics

Accessibility: The Context (Accessible Discourse Referents) as an Argument

Principles [de Groote(2006)]

$$
\begin{array}{ll}
[\![S]\!] & = \gamma \rightarrow (\gamma \rightarrow t) \rightarrow t \qquad \triangleq \Omega \\
[\![NP]\!] & = (e \rightarrow [\![S]\!]) \rightarrow [\![S]\!] \\
[\![N]\!] & = e \rightarrow [\![S]\!] \\
[\![S_1.S_2]\!] & = \lambda e.\lambda \phi.[\![S_1]\!]\ e\ (\lambda e'.[\![S_2]\!]\ e'\ \phi)
\end{array}
$$

Interpretation of sentences

$$\underbrace{\overbrace{\qquad\qquad}^{e:\gamma}\ S\ \overbrace{\qquad\qquad}^{\phi:\gamma\rightarrow t}}_{t}$$

Composition of sentences

$$\overbrace{\qquad\qquad}^{e:\gamma}\ S_1\,.\ \underbrace{S_2\ \overbrace{\qquad\qquad}^{\phi:\gamma\rightarrow t}}_{\lambda e'.[\![S_2]\!]\ e'\ \phi:\gamma\rightarrow t}$$

CS: an Example

Example

A man is sleeping.
$\lambda e.\lambda\phi.\exists x.\ (\mathbf{man}\ x) \land (\mathbf{sleeping}\ x) \land (\phi\ (x :: e))$

He is snoring.
$\lambda e.\lambda\phi.(\mathbf{snoring}\ (\mathtt{sel}\ e)) \land (\phi\ e)$

Composition of sentences

$$[\![T.S]\!] = \lambda e.\lambda\phi.[\![T]\!]\ e\ (\lambda e'.[\![S]\!]\ e'\ \phi)$$

$$\begin{aligned}
&\lambda e\ \phi.[\lambda e\ \phi.\exists x.(\mathbf{man}\ x) \land (\mathbf{sleeping}\ x) \land \phi\ (x :: e)]\ e \\
&\qquad\qquad (\lambda e'.(\lambda e\ \phi.(\mathbf{snoring}\ (\mathtt{sel}\ e)) \land (\phi\ e))\ e'\ \phi) \\
\rightarrow_\beta\quad &\lambda e\ \phi.[\lambda\phi.\exists x.(\mathbf{man}\ x) \land (\mathbf{sleeping}\ x) \land (\phi\ (x :: e))] \\
&\qquad\qquad (\lambda e'.(\mathbf{snoring}\ (\mathtt{sel}\ e')) \land (\phi\ e')) \\
\rightarrow_\beta\quad &\lambda e\ \phi.[\exists x.(\mathbf{man}\ x) \land (\mathbf{sleeping}\ x) \land ((\lambda e'.(\mathbf{snoring}\ (\mathtt{sel}\ e')) \land (\phi\ e'))\ (x :: e))] \\
\rightarrow_\beta\quad &\lambda e\ \phi.[\exists x.(\mathbf{man}\ x) \land (\mathbf{sleeping}\ x) \land ((\mathbf{snoring}\ (\mathtt{sel}\ (x :: e)) \land (\phi\ (x :: e))))]
\end{aligned}$$

Accessibility

The Context (Accessible Discourse Referents) as an Argument

Composition of sentences

$$[\![S_1.S_2]\!] = \lambda e\phi.[\![S_1]\!]\, e\, (\lambda e'.[\![S_2]\!]\, e'\, \phi)$$

Example

John owns a car

j y
car y
own j y

$\lambda e\phi.\exists y.\textbf{car}\, y \wedge \textbf{own}\, \textbf{j}\, y \wedge \phi\, (y :: e)$

it

z=?

$\lambda Pe\phi.P\, (\texttt{sel}\, e)\, e\, \phi$

it is red

z
red z
z =?

$\lambda e\phi.\textbf{red}(\texttt{sel}\, e) \wedge \phi\, e$

John owns a car
it is red

j y z
car y
own j y
red z
$z = y$

$\lambda e\phi.\exists y.\textbf{car}\, y \wedge \textbf{own}\, \textbf{j}\, y \wedge \textbf{red}(\texttt{sel}\, y :: e)$
$\wedge \phi\, (y :: e)$

Lexical Semantics

Lexicon

$$
\begin{aligned}
[\![John]\!] &= \lambda Pe\phi.P\,\mathbf{j}\,e\,\phi \\
[\![owns]\!] &= \lambda OS.S(\lambda x.O(\lambda ye'\phi'.\mathbf{own}\,x\,y \wedge \phi'\,e')) \\
[\![a]\!] &= \lambda PQe\phi.\exists y.P\,y\,(y :: e)\,\phi \wedge Q\,y\,(y :: e)\,\phi \\
[\![car]\!] &= \lambda xe\phi.\mathbf{car}\,x
\end{aligned}
$$

Example

$$
\begin{aligned}
[\![a]\!]\,[\![car]\!] &= \lambda Qe\phi.\exists y.\mathbf{car}\,y \wedge Q\,y\,(y :: e)\,\phi \\
[\![owns]\!]([\![a]\!]\,[\![car]\!]) &= \lambda S.S(\lambda x.(\lambda Qe\phi.\exists y.\mathbf{car}\,y \wedge Q\,y\,(y :: e)\,\phi) \\
&\qquad (\lambda ye'\phi'.\mathbf{own}\,x\,y \wedge \phi'\,e')) \\
&= \lambda S.S(\lambda x.(\lambda e\phi.\exists y.\mathbf{car}\,y \wedge (\lambda ye'\phi'.\mathbf{own}\,x\,y \wedge \phi'\,e')\,y\,(y :: e)\,\phi)) \\
&= \lambda S.S(\lambda x.(\lambda e\phi.\exists y.\mathbf{car}\,y \wedge (\mathbf{own}\,x\,y \wedge \phi\,(y :: e)))) \\
[\![owns]\!]([\![a]\!]\,[\![car]\!])[\![John]\!] &= (\lambda Pe\phi.P\,\mathbf{j}\,e\,\phi)(\lambda x.(\lambda e\phi.\exists y.\mathbf{car}\,y \wedge (\mathbf{own}\,x\,y \wedge \phi\,(y :: e)))) \\
&= (\lambda e\phi.(\lambda x.(\lambda e\phi.\exists y.\mathbf{car}\,y \wedge (\mathbf{own}\,x\,y \wedge \phi\,(y :: e))))\,\mathbf{j}\,e\,\phi) \\
&= \lambda e\phi.\exists y.\mathbf{car}\,y \wedge (\mathbf{own}\,\mathbf{j}\,y \wedge \phi\,(y :: e))
\end{aligned}
$$

A New Dynamic Logic [de Groote(2007)]

Logical connectives

Conjunction:	$A \sqcap B \triangleq \lambda e\phi.A\ e(\lambda e'.B\ e'\phi)$
Existential quantification:	$\Sigma x.Px \triangleq \lambda e\phi.\exists x.P\,x\,(x :: e)\,\phi$
Negation:	$\tilde{}A \triangleq \lambda e\phi.\neg(A\,e\,(\lambda e.\top)) \wedge \phi\,e$
Universal quantification:	$\Pi x.Px \triangleq \tilde{}(\Sigma x.\tilde{}(P\ x))$

Example

Lexical semantics

	Montague semantics	Dynamics semantics
a, *some*	$\lambda P\ Q.\exists x.P\ x \wedge Q\ x$	$\lambda P\ Q.\Sigma x.P\ x \sqcap Q\ x$
every	$\lambda P\ Q.\forall x.P\ x \Rightarrow Q\ x$	$\lambda P\ Q.\Pi x.P\ x \sqsupset Q\ x$

Discourse

See Philippe de Groote's slide

C. Barker.
Continuations and the nature of quantification.
Natural Language Semantics, 10:211–242, 2002.

J. Bos.
Predicate logic unplugged.
In *Proceedings of the Tenth Amsterdam Colloquium*, 1995.

J. Busquets, L. Vieu, and N. Asher.
La SDRT : une approche de la cohérence du discours dans la tradition de la sémantique dynamique.
Verbum, 23(1), 2001.

P. W. Culicover and R. Jackendoff.
Simpler Syntax.
Oxford University Press, 2005.

H. B. Curry.
Some logical aspects of grammatical structure.
In R. Jakobson, editor, *Structure of Language and its Mathematical Aspects: Proceedings of the Twelfth Symposium in Applied Mathematics*, pages 56–68.
American Mathematical Society, 1961.

P. de Groote.
Towards a montagovian account of dynamics.
In M. Gibson and J. Howell, editors, *Proceedings of Semantics and Linguistic Theory (SALT) 16*, 2006.
http://elanguage.net/journals/index.php/salt/article/view/16.1/1791.

P. de Groote.
Yet another dynamic logic.
Presentation at the 4th Lambda Calculus and Formal Grammar workshop, September 18-19 2007.
http://www.loria.fr/equipes/calligramme/acg/workshops/lcfg-04/slides/lcfg04-degroote.pdf.

P. de Groote and S. Pogodalla.
On the expressive power of Abstract Categorial Grammars: Representing context-free formalisms.
Journal of Logic, Language and Information, 13(4):421–438, 2004.
doi: 10.1007/s10849-004-2114-x.
URL http://hal.inria.fr/inria-00112956.

D. Dowty.
Grammatical relations and montague grammar.
In P. Jacobson and G. K. Pullum, editors, *The Nature of Syntactic Representation*. Reidel, Dordrecht, 1982.

M. Egg.
Semantic underspecification: Concepts and applications.
Lecture at JSM'10, 2010.
http://jsm.loria.fr/jsm10/documents/lectures/egg.pdf.

C. Gardent and L. Kallmeyer.
Semantic construction in feature-based tag.
In *Proceedings of the 10th Meeting of the European Chapter of the Association for Computational Linguistics (EACL)*, 2003.

R. Jackendoff.
Foundations of Language: Brain, Meaning, Grammar, Evolution.
Oxford University Press, 2002.

L. Kallmeyer and M. Romero.
Scope and situation binding for ltag.
Research on Language and Computation, 2007.

M. Kanazawa.
Parsing and generation as datalog queries.
In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 176–183, Prague, Czech Republic, June 2007. Association for Computational Linguistics.
http://www.aclweb.org/anthology/P/P07/P07-1023.

M. Kanazawa.
A prefix-correct earley recognizer for multiple context-free grammars.
In *Proceedings of the Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 49–56, Tuebingen, Germany, June 7–8 2008.

M. Kanazawa.
Second-order abstract categorial grammars as hyperedge replacement grammars.
Journal of Logic, Language, and Information, 19(2):137–161, 2009.
http://www.springerlink.com/content/p644605651088uv6/.

M. Kanazawa and S. Salvati.
Generating control languages with abstract categorial grammars.
In G. Penn, editor, *Proceedings of The 12th conference on Formal Grammar FG 2007*. CSLI Publications, 2007.
http://www.dei.unipd.it/~fgrammar/fg07/fg07preproc/0005/paper_10.pdf.

M. Kracht.
The Mathematics of Language.
Number 63 in Studies in Generative Grammar. Mouton de Gruyter, Berlin, 2003.

R. Montague.
Universal grammar.
Theoria, 36:373–398, 1970.

R. Montague.
The proper treatment of quantification in ordinary english.
In *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, 1974.
Re-edited in "Formal Semantics: The Essential Readings", Paul Portner and Barbara H. Partee, editors. Blackwell Publishers, 2002.

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.

R. Muskens.
Lambdas, Language, and Logic.
In G.-J. Kruijff and R. Oehrle, editors, *Resource Sensitivity in Binding and Anaphora*, Studies in Linguistics and Philosophy, pages 23–54. Kluwer, 2003.

D. Oehrle.
Some 3-Dimensional Systems of Labelled Deduction.
Logic Jnl IGPL, 3(2-3):429–448, 1995.
doi: 10.1093/jigpal/3.2-3.429.
URL http://jigpal.oxfordjournals.org.

R. T. Oehrle.
Term-labeled categorial type systems.
Linguistic and Philosophy, 17(6):633–678, December 1994.
doi: 10.1007/BF00985321.

C. Pollard.
High-order categorical grammars.
In *Proceedings of Categorial Grammars 2004, Montpellier*, pages 340–361, June 2004.

C. Pollard.
An introduction to convergent grammar.
Presentation at Calligramme Seminar, 2008.
http://www.ling.ohio-state.edu/~scott/cvg/.

A. Ranta.
Type Theoretical Grammar.
Oxford University Press, 1994.

S. Salvati.
Encoding second order string ACG with Deterministic Tree Walking Transducers.
In Shuly Wintner, editor, *Proceedings of The 11th conference on Formal Grammar FG 2006*, FG Online Proceedings, pages 143–156, Malaga Espagne, 2006. CSLI Publications.
http://cslipublications.stanford.edu/FG/2006/salvati.pdf.

S. Salvati.
On the complexity of Abstract Categorial Grammars.
In *Proceedings of the 10th Conference on Mathematics of Language, MOL 10*, 2007.
http://wwwhomes.uni-bielefeld.de/mkracht/mol10/abstracts/acg_complexity.pdf.

D. J. Weir.
Characterizing Mildly Context-Sensitive Grammar Formalisms.
PhD thesis, University of Pennsylvania, 1988.

Y. Winter.
Elements of Formal Semantics.
To appear in Edimburgh University Press, 2010.