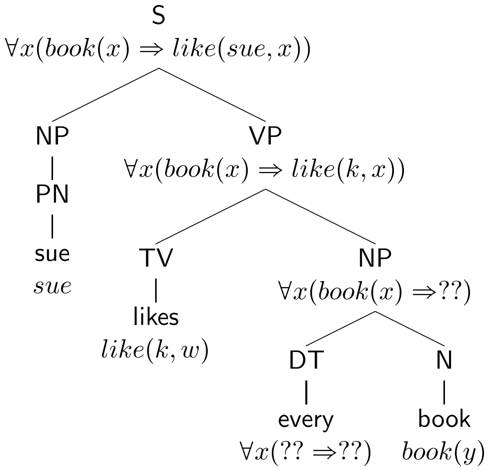
Computational Semantics: Syntax-Semantics Interface

Chapter 18 J&M'09

Compositionality

How can we automate the process of associating semantic representations with natural language expressions?

Is this process systematic?



Compositionality

We'll extend FOL with $\lambda-$**calculus** :

- If ϕ is a formula and v is a variable, then $\lambda v.\phi$ is a formula
- If $\lambda v.\phi$ is a formula, then $(\lambda v.\phi)(\psi)$ is a formula.

Examples of well-formed expressions:
$\lambda x.sneeze(x)$
$(\lambda x.sneeze(x)(sue))$

In general, $\lambda v.\phi$ is a function. It takes some argument ψ, and outputs ϕ after replacing all occurrences of v in ϕ by ψ.

Formally, this is written: $(\lambda v.\phi)(\psi) = \phi[v/\psi]$

This $\phi[v/\psi]$ operation is called β-**reduction**. Example:

$(\lambda x.sneeze(x))(sue) = sneeze(sue)$ via $[x/sue]$

Compositionality

Our $\lambda-$terms will employ two kinds of variable:

- Variables over individuals: $x_1 ..., x_n, y_1, ...y_n, ...$
- Variable over other λ-terms: $P, Q, R, T, K, ...$

Examples:

- $(\lambda x.sneeze(x))(sue) = sneeze(sue)$
- $(\lambda x.sneeze(x))(\lambda y.sigh(y)) \neq sneeze(\lambda y.sigh(y))$
  Because $(\lambda x.sneeze(x))(\lambda y.sigh(y))$ is not well-formed.
- $(\lambda P.P(x) \wedge house(x))(\lambda y.yellow(y)) =$
  $(\lambda y.yellow(y))(x) \wedge house(x) = yellow(x) \wedge house(x)$

Compositionality

Extending a grammar with semantic representations:

CFG nodes are now of the form: (Category ; FOL)

- **Proper names:** (not final version)

  (PN ; tom) → *tom*

  (PN ; mia) → *mia*

  ...
- **Intransitive verbs:**

  (IV ; $\lambda x.sneeze(x)$) → *sneezed*

  (IV ; $\lambda x.snore(x)$) → *snores*
- **transitive verbs:**

  (TV ; $\lambda y.\lambda x.like(x,y)$) → *likes* (not final version)

  ...
- **Phrasal rules:**

  (NP; ϕ) → (PN; ϕ)

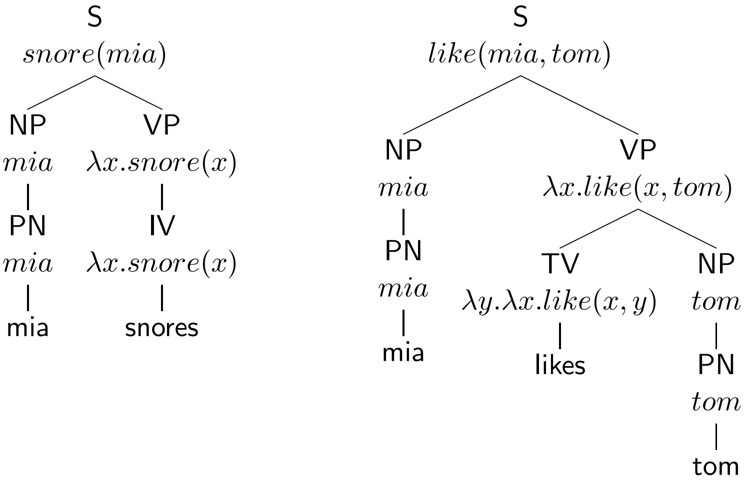
  (VP; ϕ) → (IV; ϕ)

  (VP; $(\phi)(\psi)$) → (TV; ϕ) (NP; ψ)

  (S; $(\psi)(\phi)$) → (NP; ϕ) (VP; ψ) (not final version)

Compositionality

This grammar fragment licenses the following:



Complex NPs:

- **Determiners:**
  (DT ; $\lambda P.\lambda Q.\forall x((P)(x) \Rightarrow (Q)(x))$) $\rightarrow$ *every* | *each* | *all*
  (DT ; $\lambda P.\lambda Q.\exists x((P)(x) \wedge (Q)(x))$) $\rightarrow$ *a* | *an* | *some*
  (DT ; $\lambda P.\lambda Q.\neg\exists x((P)(x) \wedge (Q)(x))$) $\rightarrow$ *no*
  ...
- **Common nouns:**
  (N ; $\lambda x.cat(x)$) $\rightarrow$ *cat*
  (N ; $\lambda x.dog(x)$) $\rightarrow$ *dog*
  ...
- **More phrasal rules:**
  (NP; $(\phi)(\psi)$) $\rightarrow$ (DT; ϕ) (N; ψ)

Compositionality

To model complex NP subjects, we need to revise the S rule:
(S; $(\phi)(\psi)$) $\rightarrow$ (NP; ϕ) (VP; ψ)

S

$$(\lambda Q.\forall y(cat(y) \Rightarrow Q(y)))(\lambda w.snore(w)) =$$
$$\forall y(cat(y) \Rightarrow (\lambda w.snore(w))(y)) =$$
$$\forall y(cat(y) \Rightarrow snore(y))$$

NP

$$(\lambda P.\lambda Q.\forall y(P(y) \Rightarrow Q(y)))(\lambda x.cat(x)) =$$
$$\lambda Q.\forall y((\lambda x.cat(x))(y) \Rightarrow Q(y)) =$$
$$\lambda Q.\forall y(cat(y) \Rightarrow Q(y))$$

DT

$$\lambda P.\lambda Q.\forall y(P(y) \Rightarrow Q(y))$$

every

N

$$\lambda x.cat(x)$$

VP

$$\lambda w.snore(w)$$

IV

$$\lambda w.snore(w)$$

snores

Revising PN and TV representations accordingly:

- **Proper names:** (final version)
  (PN ; $\lambda P.(P)(tom)$) → *tom*
  (PN ; $\lambda P.(P)(mia)$) → *mia*
  ...
- **Verbs:**
  (IV ; $\lambda x.sneeze(x)$) → *sneezed*
  (TV ; $\lambda R.\lambda x.(R)(\lambda y.like(x, y))$) → *likes*
  ...
- **Phrasal rules:**
  (VP; $(\phi)(\psi)$) → (TV; ϕ) (NP; ψ)
  (S; $(\phi)(\psi)$) → (NP; ϕ) (VP; ψ)

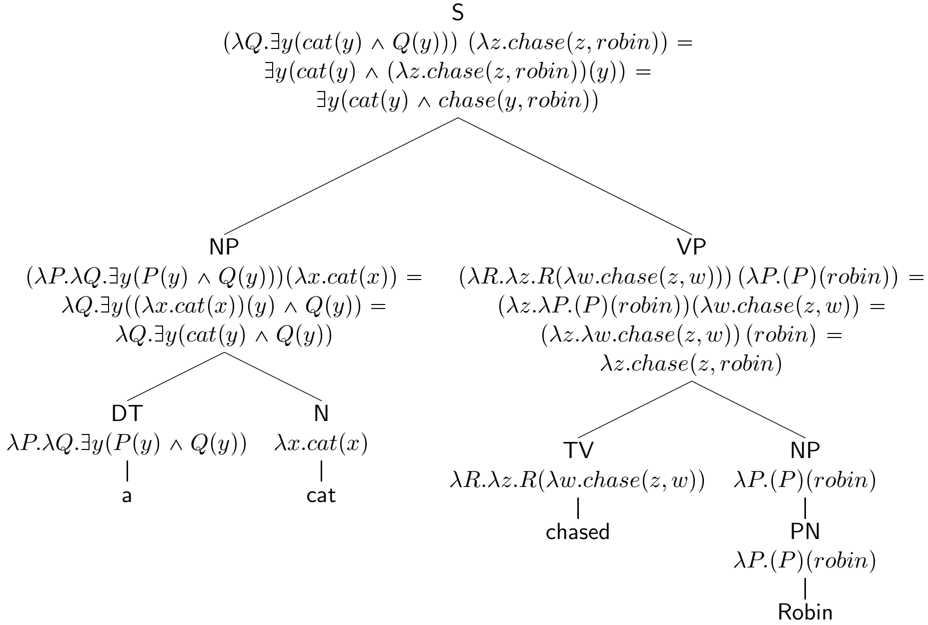
Compositionality

S

$$(\lambda P.(P)(tom))(\lambda x.sneeze(x)) =$$
$$(\lambda x.sneeze(x))(tom) =$$
$$sneeze(tom)$$

NP	VP
$\lambda P.(P)(tom)$	$\lambda x.sneeze(x)$
PN	IV
$\lambda P.(P)(tom)$	$\lambda x.sneeze(x)$
Tom	sneezed

- S
  $$(\lambda P.(P)(tom))\ (\lambda z.\forall y(cat(y) \Rightarrow like(z,y))) =$$
  $$(\lambda z.\forall y(cat(y) \Rightarrow like(z,y)))\ (tom) =$$
  $$\forall y(cat(y) \Rightarrow like(tom,y))$$
  - NP
    $\lambda P.(P)(tom)$
    - PN
      $\lambda P.(P)(tom)$
      - Tom
  - VP
    $$(\lambda R.\lambda z.R(\lambda w.like(z,w)))\,(\lambda Q.\forall y(cat(y) \Rightarrow Q(y))) =$$
    $$(\lambda z.\lambda Q.\forall y(cat(y) \Rightarrow Q(y)))\ (\lambda w.like(z,w)) =$$
    $$\lambda z.\forall y(cat(y) \Rightarrow (\lambda w.like(z,w))(y)) =$$
    $$\lambda z.\forall y(cat(y) \Rightarrow like(z,y))$$
    - TV
      $\lambda R.\lambda z.R(\lambda w.like(z,w))$
      - likes
    - NP
      $$(\lambda P.\lambda Q.\forall y(P(y) \Rightarrow Q(y)))(\lambda x.cat(x)) =$$
      $$\lambda Q.\forall y((\lambda x.cat(x))(y) \Rightarrow Q(y)) =$$
      $$\lambda Q.\forall y(cat(y) \Rightarrow Q(y))$$
      - DT
        $\lambda P.\lambda Q.\forall y(P(y) \Rightarrow Q(y))$
        - every
      - N
        $\lambda x.cat(x)$
        - cat


S
$(\lambda Q.\exists y(cat(y) \wedge Q(y)))\ (\lambda z.chase(z, robin)) =$
$\exists y(cat(y) \wedge (\lambda z.chase(z, robin))(y)) =$
$\exists y(cat(y) \wedge chase(y, robin))$
NP
$(\lambda P.\lambda Q.\exists y(P(y) \wedge Q(y)))(\lambda x.cat(x)) =$
$\lambda Q.\exists y((\lambda x.cat(x))(y) \wedge Q(y)) =$
$\lambda Q.\exists y(cat(y) \wedge Q(y))$
VP
$(\lambda R.\lambda z.R(\lambda w.chase(z, w)))\,(\lambda P.(P)(robin)) =$
$(\lambda z.\lambda P.(P)(robin))(\lambda w.chase(z, w)) =$
$(\lambda z.\lambda w.chase(z, w))\,(robin) =$
$\lambda z.chase(z, robin)$
DT
$\lambda P.\lambda Q.\exists y(P(y) \wedge Q(y))$
a
N
$\lambda x.cat(x)$
cat
TV
$\lambda R.\lambda z.R(\lambda w.chase(z, w))$
chased
NP
$\lambda P.(P)(robin)$
PN
$\lambda P.(P)(robin)$
Robin

Compositionality

S

$$(\lambda Q.\forall y(cat(y) \Rightarrow Q(y)))(\lambda w.yawn(w)) =$$
$$\forall y(cat(y) \Rightarrow (\lambda w.yawn(w))(y)) =$$
$$\forall y(cat(y) \Rightarrow yawn(y))$$

NP

$$(\lambda P.\lambda Q.\forall y(P(y) \Rightarrow Q(y)))(\lambda x.cat(x)) =$$
$$\lambda Q.\forall y((\lambda x.cat(x))(y) \Rightarrow Q(y)) =$$
$$\lambda Q.\forall y(cat(y) \Rightarrow Q(y))$$

VP

$$\lambda w.yawn(w)$$

IV

$$\lambda w.yawn(w)$$

DT

$$\lambda P.\lambda Q.\forall y(P(y) \Rightarrow Q(y))$$

N

$$\lambda x.cat(x)$$

- S
  $\lambda Q.\forall y(dog(y) \Rightarrow Q(y))\ (\lambda z.\exists k(cat(k) \wedge chase(z,k))) =$
  $\forall y(dog(y) \Rightarrow \lambda z.\exists k(cat(k) \wedge chase(z,k))(y)) =$
  $\forall y(dog(y) \Rightarrow \exists k(cat(k) \wedge chase(y,k)))$
  - NP
    $\lambda P.\lambda Q.\forall y(P(y) \Rightarrow Q(y))(\lambda x.dog(x)) =$
    $\lambda Q.\forall y(\lambda x.dog(x)(y) \Rightarrow Q(y)) =$
    $\lambda Q.\forall y(dog(y) \Rightarrow Q(y))$
    - DT
      $\lambda P.\lambda Q.\forall y(P(y) \Rightarrow Q(y))$
      - every
    - N
      $\lambda x.dog(x)$
      - dog
  - VP
    $\lambda R.\lambda z.R(\lambda w.chase(z,w))\ \lambda Q'.\exists k(cat(k) \wedge Q'(k)) =$
    $\lambda z.\lambda Q'.\exists k(cat(k) \wedge Q'(k))(\lambda w.chase(z,w)) =$
    $\lambda z.\exists k(cat(k) \wedge \lambda w.chase(z,w)(k)) =$
    $\lambda z.\exists k(cat(k) \wedge chase(z,k))$
    - TV
      $\lambda R.\lambda z.R(\lambda w.chase(z,w))$
      - chased
    - NP
      $\lambda P'.\lambda Q'.\exists k(P(k) \wedge Q'(k))(\lambda y.cat(y))$
      $\lambda Q'.\exists k(\lambda y.cat(y)(k) \wedge Q'(k)) =$
      $\lambda Q'.\exists k(cat(k)\ \wedge\ Q'(k))$
      - DT
        $\lambda P'.\lambda Q'.\exists k(P'(k) \wedge Q'(k))$
        - a
      - N
        $\lambda y.cat(y)$
        - cat

Compositionality

Lexicon sample:

- (PN ; $\lambda P.(P)(tom)$) $\rightarrow$ *tom*
- (N ; $\lambda y.cat(y)$) $\rightarrow$ *cat*
- (DT ; $\lambda P.\lambda Q.\forall x((P)(x) \Rightarrow (Q)(x))$) $\rightarrow$ *every* | *each* | *all*
- (DT ; $\lambda P.\lambda Q.\exists x((P)(x) \wedge (Q)(x))$) $\rightarrow$ *a* | *an* | *some*
- (DT ; $\lambda P.\lambda Q.\neg\exists x((P)(x) \wedge (Q)(x))$) $\rightarrow$ *no*
- (IV ; $\lambda x.sneeze(x)$) $\rightarrow$ *sneezed*
- (TV ; $\lambda R.\lambda x.(R)(\lambda y.like(x,y))$) $\rightarrow$ *likes*

Grammar rules, so far:

- (NP; ϕ) $\rightarrow$ (PN; ϕ)
- (VP; ϕ) $\rightarrow$ (IV; ϕ)
- (NP; $(\phi)(\psi)$) $\rightarrow$ (DT; ϕ) (N; ψ)
- (VP; $(\phi)(\psi)$) $\rightarrow$ (TV; ϕ) (NP; ψ)
- (S; $(\phi)(\psi)$) $\rightarrow$ (NP; ϕ) (VP; ψ)

Adnominal modifiers

- **Adjectives**
  (ADJ ; $\lambda P.\lambda x.((P)(x) \wedge yellow(x))$) → *yellow*
  (ADJ ; $\lambda P.\lambda x.((P)(x) \wedge black(x))$) → *black*
  (ADJ ; $\lambda P.\lambda x.((P)(x) \wedge happy(x))$) → *happy*
- **Phrasal Rule:**
  (N ; $(\phi)(\psi)$) → (ADJ ; ϕ) (N ; ψ)

N

$$(\lambda P.\lambda x.((P)(x) \wedge yellow(x)))(\lambda y.cat(y)) =$$
$$\lambda x.((\lambda y.cat(y))(x) \wedge yellow(x)) =$$
$$\lambda x.(cat(x) \wedge yellow(x))$$

ADJ $\lambda P.\lambda x.((P)(x) \wedge yellow(x))$ — yellow

N $\lambda y.cat(y)$ — cat

Compositionality

Adnominal modifiers (continued)

- **Prepositions**
  (P ; $\lambda P.\lambda Q.\lambda x.((Q)(x) \wedge (P)(\lambda y.in(x,y)))$) $\rightarrow$ *in*
  (P ; $\lambda P.\lambda Q.\lambda x.((Q)(x) \wedge (P)(\lambda y.on(x,y)))$) $\rightarrow$ *on*
  (P ; $\lambda P.\lambda Q.\lambda x.((Q)(x) \wedge (P)(\lambda y.near(x,y)))$) $\rightarrow$ *near*
  (P ; $\lambda P.\lambda Q.\lambda x.((Q)(x) \wedge (P)(\lambda y.under(x,y)))$) $\rightarrow$ *under*
  (P ; $\lambda P.\lambda Q.\lambda x.((Q)(x) \wedge (P)(\lambda y.over(x,y)))$) $\rightarrow$ *over*
- **Phrasal rules:**
  (PP; $(\phi)(\psi)$) $\rightarrow$ (P; ϕ) (NP; ψ)
  (N; $(\psi)(\phi)$) $\rightarrow$ (N; ϕ) (PP; ψ)

Example:

$(\lambda P.\lambda Q.\lambda x.((Q)(x) \wedge (P)(\lambda y.in(x,y))))\ (\lambda W.(W)(paris)) =$
$(\lambda Q.\lambda x.((Q)(x) \wedge (\lambda W.(W)(paris))(\lambda y.in(x,y)))) =$
$(\lambda Q.\lambda x.((Q)(x) \wedge (\lambda y.in(x,y))(paris))) =$
$\lambda Q.\lambda x.((Q)(x) \wedge in(x,paris))$

Compositionality

Big picture

- The meaning of an expression is a function of the meanings of its parts and the way they are syntactically combined.
- The order of functional composition is dictated by the information contained in lexical representations and by language-specific phrasal rules.
- $\lambda-$terms can be used to formalize a very wide range of linguistic patterns, following the same rationale.