Introduction to Formal Semantics: RECAP

Volha Petukhova & Nicolaie Dominik Dascalu
Spoken Language Systems Group
Saarland University
11.07.2022
Overview for today

- All 10 topics: summary of important issues
- Question session
- Some formalities
Meaning representation

- Structures from set of symbols
  - representational vocabulary

- Symbol structures correspond to:
  - objects
  - properties of objects
  - relations among objects

- Can be viewed as:
  - representation of meaning of linguistic input
  - representation of state of world
A model is an abstract mathematical structure that we construct for describing hypothetical situations. Models are used for analyzing natural language expressions (words, phrases and sentences) by associating them with abstract objects.

A semantic theory $T$ satisfies the **truth-conditional criterion** (TCC) for sentences $S_1$ and $S_2$ if the following two conditions are equivalent:

1. Sentence $S_1$ intuitively entails $S_2$.
2. For all models $M$ in $T$: $[S_1]^M \leq [S_2]^M$
## Topic 1: Meaning & Form

Models: truth values

1. Tina is tall and thin.
2. Tina is tall.
1. $\implies$ 2

<table>
<thead>
<tr>
<th>Expression</th>
<th>Cat.</th>
<th>Type</th>
<th>Abstract denotation</th>
<th>Denotations in example models with $E = {a, b, c, d}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tina</td>
<td>PN</td>
<td>entity</td>
<td>tina</td>
<td>$a$</td>
</tr>
<tr>
<td>tall</td>
<td>A</td>
<td>set of entities</td>
<td>tall</td>
<td>${b,c}$</td>
</tr>
<tr>
<td>thin</td>
<td>A</td>
<td>set of entities</td>
<td>thin</td>
<td>${b,d}$</td>
</tr>
<tr>
<td>tall and thin</td>
<td>AP</td>
<td>set of entities</td>
<td>AND(tall, thin)</td>
<td>${a,b,c}$</td>
</tr>
<tr>
<td>Tina is thin</td>
<td>S</td>
<td>truth-value</td>
<td>IS(tina, thin)</td>
<td>${b,c}$</td>
</tr>
<tr>
<td>Tina is tall and thin</td>
<td>S</td>
<td>truth-value</td>
<td>IS(tina, AND(tall, thin))</td>
<td>${b}$</td>
</tr>
</tbody>
</table>

Introduction into Formal Semantics, Summer 2021
Compositionality: the denotation of a complex expressions is determined by the denotations of its immediate parts and the ways they combine with each other
Topic 2: Inference

$p$ entails $q$

- whenever $p$ is true, $q$ must be true
- a situation describable by $p$ must be also describable by $q$
- $p$ and $\neg q$ is contradictory (cannot be true in any situation)

Entailment: (i) indefeasible; (ii) speakers intuitively accept S2 whenever they accept S1.

Presupposition a special type of entailment

- $p$ entails $q$ iff $q$ is true whenever $p$ is true
- $p$ presupposes $q$ iff $q$ is true whenever $p$ or $\neg p$ is true

Presupposition projection (tests). Presupposition are defeasible (contextual and structural defeasibility; examples).
Presuppositions are special entailments. Ordinary entailments disappear when the sentence is *negated*, turned into a *question*, or embedded under a *possibility modal* like maybe:

Example

<table>
<thead>
<tr>
<th>Alice has a cute dog.</th>
<th>⇒</th>
<th>Alice has a dog.</th>
</tr>
</thead>
<tbody>
<tr>
<td>b. Alice doesn’t have a cute dog.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c. Does Alice have a cute dog?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d. Maybe Alice has a cute dog.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b–d:</td>
<td></td>
<td>¬⇒</td>
</tr>
</tbody>
</table>
Presuppositions are special entailments. Ordinary entailments disappear when the sentence is *negated*, turned into a *question*, or embedded under a *possibility modal* like maybe:

**Example**

1. Alice has a cute dog.
Presupposition (cont.)

Presuppositions are special entailments. Ordinary entailments disappear when the sentence is *negated*, turned into a *question*, or embedded under a *possibility modal* like maybe:

**Example**

1. Alice has a cute dog.
2. $\Rightarrow$ Alice has a dog.
Presupposition (cont.)

Presuppositions are special entailments. Ordinary entailments disappear when the sentence is *negated*, turned into a *question*, or embedded under a *possibility modal* like maybe:

**Example**

1. Alice has a cute dog.
2. $\implies$ Alice has a dog.
3. b. Alice doesn’t have a cute dog.
Presupposition (cont.)

Presuppositions are special entailments. Ordinary entailments disappear when the sentence is negated, turned into a question, or embedded under a possibility modal like maybe:

Example

1. Alice has a cute dog.
2. Alice has a dog.
3. b. Alice doesn’t have a cute dog.
4. c. Does Alice have a cute dog?
Presuppositions are special entailments. Ordinary entailments disappear when the sentence is negated, turned into a question, or embedded under a possibility modal like maybe:

Example

1. Alice has a cute dog.
2. \(\Rightarrow\) Alice has a dog.
3. b. Alice doesn’t have a cute dog.
4. c. Does Alice have a cute dog?
5. d. Maybe Alice has a cute dog.
Presuppositions are special entailments. Ordinary entailments disappear when the sentence is *negated*, turned into a *question*, or embedded under a *possibility modal* like maybe:

### Example

1. Alice has a cute dog.
2. \(\implies\) Alice has a dog.
3. b. Alice doesn’t have a cute dog.
4. c. Does Alice have a cute dog?
5. d. Maybe Alice has a cute dog.
6. (b–d): \(\nleftrightarrow\) Alice has a dog.
Presupposition (cont.)

But presuppositions survive in all of these environments (we also say: presuppositions project):

Example

1. Bob's sister is a professional wrestler.
2. \( \Rightarrow \)
3. Bob has a sister.
4. b. Bob's sister is not a professional wrestler.
5. c. Is Bob's sister a professional wrestler?
6. d. Maybe Bob's sister is a professional wrestler.

(b–d): \( \Rightarrow \)

\{o.petukhova; n.dascalu\}@lsv.uni-saarland.de

Introduction into Formal Semantics, Summer 2021
But presuppositions survive in all of these environments (we also say: presuppositions project):

Example

1. Bob’s sister is a professional wrestler.
Presupposition (cont.)

But presuppositions survive in all of these environments (we also say: presuppositions project):

Example

1. Bob’s sister is a professional wrestler.
2. $\Rightarrow$ Bob has a sister.
Presupposition (cont.)

But presuppositions survive in all of these environments (we also say: presuppositions project):

Example

1. Bob’s sister is a professional wrestler.
2. $\Rightarrow$ Bob has a sister.
3. b. Bob’s sister is not a professional wrestler.
Presupposition (cont.)

But presuppositions survive in all of these environments (we also say: presuppositions project):

Example

1. Bob’s sister is a professional wrestler.
2. $\Rightarrow$ Bob has a sister.
3. b. Bob’s sister is not a professional wrestler.
4. c. Is Bob’s sister a professional wrestler?
Presupposition (cont.)

But presuppositions survive in all of these environments (we also say: presuppositions project):

Example

1. Bob’s sister is a professional wrestler.
2. $\Rightarrow$ Bob has a sister.
3. b. Bob’s sister is not a professional wrestler.
4. c. Is Bob’s sister a professional wrestler?
5. d. Maybe Bob’s sister is a professional wrestler.
Presupposition (cont.)

But presuppositions survive in all of these environments (we also say: presuppositions project):

Example

1. Bob’s sister is a professional wrestler.
2. Bob has a sister.
3. Bob’s sister is not a professional wrestler.
4. Is Bob’s sister a professional wrestler?
5. Maybe Bob’s sister is a professional wrestler.
6. (b–d): Bob has a sister.
Problem 1 Presupposition failure (= the presupposition is false in context)

(1) King of France is bald.
   \textit{When uttered on May 13 2005, the presupposition is false}

Problem 2 Presupposition cancellation (= the presupposition is “removed” in context)

(2) I don’t know that Bill came.
   \textit{This utterance does not presuppose that speaker knows that Bill came.}
(3) The king of France is bald.

Russell:

• is quantificational. It is logically equivalent to: “exactly one entity has the property King of France, and that entity is bald”
• Thus, if there is no unique King of France, (3) is false.
• \( \exists x. KOF = \{x\} \land \text{bald}(x) \)

Strawson:

• Any use of (3) raises the following presupposition: “exactly one entity, call it x, has the property King of France”
• Under this presupposition, (3) means: “x is bald”
• Thus, if there is no unique King of France, (3) is neither true nor false.
• \( \exists x. KOF = \{x\} : \exists x KOF = \{x\} \land \text{bald}(x) \)
Topic 2: Inference (Trivalent Strawsonian semantics)

\[[S]^M = \phi : \psi\]

\(\phi\) indicates whether \(S\) is assertible in \(M\)
\(\psi\) indicates whether \(S\) is true in \(M\)

Definition: the colon operator (‘transplication’):

\[(\phi : \psi) = \begin{cases} 
\psi & \phi = 1 \\
* & \phi = 0 
\end{cases}\]

where \(\phi\) and \(\psi\) are bivalent truth-values.

Blamey (1986) suggests the possibility of reading the transplication connective as a type of conditionals of the form ‘if \(\phi\) then \(\psi\)’ which are neither true nor false when \(\phi\) is false. They are also neither true nor false when either \(\phi\) or \(\psi\) is neither true nor false.
Topic 3: Predication & Argumentation

Anatomy of First-Order-Logic (syntax and semantics): variables, constants, predicates, functions, logical connectives, quantifiers
Predicate Logics: well-formed formulas, scope and binding
Able to translate to FOL and to NL
Topic 4: Typed Lambda (abstraction from fully specified FOL)

Example

John loves Mary

$Loves(j, m)$

$Loves(_, m)$

to abstract OVER the missing piece, ABSTRACTION OPERATOR $\lambda$ is used

Example

John loves Mary

$Loves(j, m)$

$Loves(_, m)$

$\lambda x. Loves(x, m)$

This expression denotes a function from an individual to truth-value
Syntactic categories of languages $L_{\text{Pred}}$ are terms, predicates and formulas.

Language $L_{\lambda}$ has a set of TYPES which are recursively specified (of arbitrary complexity and depth), with two BASIC TYPES:

- **e** entities for individuals corresponding to terms in $L_{\text{Pred}}$
- **t** truth values for formulas

and FUNCTION TYPES: $<e, t>$ denoting functions from individuals to truth values.

A set of types is defined recursively:

- **e** is a type
- **t** is a type
- if **σ** is a type and **τ** is a type then $<\sigma, \tau>$ is a type
- nothing else is a type
### Example

\[
\lambda x. \lambda y. \text{Near}(x, y) \\
\lambda x. \lambda y. \text{Near}(x, y)(\text{midway}) \\
\lambda y. \text{Near}(\text{midway}, y) \\
\lambda y. \text{Near}(\text{midway}, y)(\text{chicago}) \\
\text{Near}(\text{midway}, \text{chicago})
\]
∀x.[Person(x) → Loves(x, m)]
Topic 4: Typed Lambda (trees)

$\text{S}$

$t$

$\text{in}(\text{joe, texas})$

$\text{NP}$

$\text{e}$

$\text{joe}$

$\text{VP}$

$\text{et}$

$\lambda y.\text{in}(y, \text{texas})$

$\text{PP}$

$\langle \text{et, et} \rangle$

$\lambda f. f$

$\lambda y.\text{in}(y, \text{texas})$

$\text{NP}$

$\text{e}$

$\langle \text{et, et} \rangle$

$\lambda x. [\lambda y.\text{in}(y, x)]$

$\text{P}$

$\langle \text{e, et} \rangle$

$\lambda x. [\lambda y.\text{in}(y, x)]$

$\text{NP}$

$\text{e}$

$\langle \text{et, et} \rangle$

$\lambda x. [\lambda y.\text{in}(y, x)]$

$\text{N}$

$\text{e}$

$\langle \text{et, et} \rangle$

$\lambda x. [\lambda y.\text{in}(y, x)]$

$\text{N}$

$\text{e}$

$\langle \text{et, et} \rangle$

$\lambda x. [\lambda y.\text{in}(y, x)]$

$\text{tx}$

$\text{texas}$

$\text{tx}$

$\text{texas}$

$\text{tx}$

$\text{texas}$

$\text{tx}$

$\text{texas}$

$\text{tx}$

$\text{texas}$

$\text{tx}$

$\text{texas}$

$\text{tx}$

$\text{texas}$
Composition Rule 1: Function Application

Let $\gamma$ be a syntax tree whose sub-trees are $\alpha$ and $\beta$ where:
- $\alpha \rightsquigarrow \alpha'$ where $\alpha'$ has type $\langle \sigma, \tau \rangle$
- $\beta \rightsquigarrow \beta'$ where $\beta'$ has type $\langle \sigma \rangle$

then $\gamma \rightsquigarrow \alpha'(\beta')$
Composition Rule 2: Non-branching Nodes

If $\beta$ is a tree whose only daughter is $\alpha$, where $\alpha \leadsto \alpha'$ then $\beta \leadsto \alpha'$

$$
\begin{array}{c}
S \\
\downarrow \\
\downarrow \\
\text{Laughs}(li) \\
\downarrow \\
\downarrow \\
\text{PN} \quad \text{VP} \\
\quad \text{e} \quad \langle e, t \rangle \\
\quad \text{li} \quad \lambda x.\text{Laughs}(x) \\
\text{Lisa} \quad \text{V} \\
\quad \langle e, t \rangle \\
\quad \lambda x.\text{Laughs}(x) \\
\quad \text{laughs}
\end{array}
$$
Topic 5: Function Application (compositionality)

- transitive verbs
- identity function
- negation
- indefinite article
- prepositional phrases
- quantification
- generalised quantifiers
Solutions for Adjectives:

(i) generate two translations: one of $\langle e, t \rangle$ for predicative positions and $\langle \langle e, t \rangle, \langle e, t \rangle \rangle$ for attributive positions;

(ii) introduce a new composition rule and give intersective adjectives one single translation

for (i) take one translation to be basic and derive the other one from it with the help of either

a TYPE-SHIFTING RULE (invisible to the syntactic component of the grammar) or

a SILENT OPERATOR (a reflection in the syntax)
Topic 6: Beyond Function Application (Silent Operator)

\[
\begin{align*}
A & \\
\langle\langle e, t\rangle, \langle e, t\rangle\rangle & \\
\lambda P. \lambda x. [Reasonable(x) \land P(x)] & \\
\langle\langle e, t\rangle, \langle\langle e, t\rangle, \langle e, t\rangle\rangle\rangle & \\
\lambda P'. \lambda P. \lambda x. [P'(x) \land P(x)] & \\
\text{MOD} & \\
\lambda x. Reasonable(x) & \\
\langle e, t\rangle & \\
\text{reasonable} & 
\end{align*}
\]
Type-Shifting Rule 1: Predicate-to-modifier shift

If $\alpha \leadsto \alpha'$, where $\alpha'$ is of type $\langle e, t \rangle$, then $\alpha \leadsto \lambda P.[\alpha'(x) \land P(x)]$ (as long as $P$ and $x$ are not free in $\alpha$; in that case, use different variables of the same type).

$$
\begin{array}{c}
A \\
\langle \langle e, t \rangle, \langle e, t \rangle \rangle \\
\lambda P. \lambda x.[Reasonable(x) \land P(x)] \\
\uparrow_{MOD} \\
A \\
\langle e, t \rangle \\
\lambda x. Reasonable(x) \\
\mid \\
\text{reasonable}
\end{array}
$$

Topic 6: Beyond Function Application (Type Shifting)

Type-Shifting Rule 1: Predicate-to-modifier shift

If $\alpha \leadsto \alpha'$, where $\alpha'$ is of type $\langle e, t \rangle$, then $\alpha \leadsto \lambda P.[\alpha'(x) \land P(x)]$ (as long as $P$ and $x$ are not free in $\alpha$; in that case, use different variables of the same type).

$$
\begin{array}{c}
A \\
\langle \langle e, t \rangle, \langle e, t \rangle \rangle \\
\lambda P. \lambda x.[Reasonable(x) \land P(x)] \\
\uparrow_{MOD} \\
A \\
\langle e, t \rangle \\
\lambda x. Reasonable(x) \\
\mid \\
\text{reasonable}
\end{array}
$$

(ii) assumption is that all intersective adjectives have translations of a single type, and we eliminate type mismatches via a new composition rule

\[
\text{NP} \\
\langle e, t \rangle \\
\lambda x. [\text{Reasonable}(x) \land \text{Doubt}(x)]
\]

\[
\begin{array}{c}
\text{AP} \\
\langle e, t \rangle \\
\lambda x. \text{Reasonable}(x)
\end{array} \\
\begin{array}{c}
\text{NP} \\
\langle e, t \rangle \\
\lambda x. \text{Doubt}(x)
\end{array}
\]


reasonable doubt
Florian vs Sandra: the successful author from India

a. DP
   | D
   | the
   | NP
   | AP  
   | successful
   | NP'
   | author
   | PP
   | from
   | N
   | India

b. DP
   | D
   | the
   | NP'
   | N
   | PP
   | AP
   | successful
   | N
   | P
   | from
   | N
   | India
Florian vs Sandra: *the successful author from India*

a. DP
   └── NP
       └── AP
           └── NP'
               └── N
                   └── PP
                       └── P
                           └── N

successful

DP
   └── D
       └── NP
           └── the

Author

from

India
Florian vs Sandra: *the successful author from India*

a. DP

```
  (DP
   (D)
   (NP
     (the)
     (AP
       (successful)
       (NP'
         (N)
         (PP
           (P
             (from)
             (N
               (e)
               (i)
               (N
                 (from)
                 (N
                   (i)
                   (N
                     (i)
                     (India)

```
Florian vs Sandra: the successful author from India

```
the (NP' successful)

successful (NP' author)

author (P [from (x, y)])

from (PP [from (x, y)])
```

[DP]

```
the (NP'

successful (NP'

author (P [from (x, y)]))

from (PP [from (x, y)])
```

[DP]

```
the (NP

successful (NP

author (P [from (x, y)]))

from (PP [from (x, y)])
```

[DP]
Florian vs Sandra: the successful author from India
Florian vs Sandra: the successful author from India

\[ \text{DP} \]

\[ \text{NP} \]

\[ \text{the} \]

\[ \text{AP} \]

\[ \text{NP'} \]

\[ \text{successful} \]

\[ \text{N} \]

\[ \lambda x. \text{Author}(x) \]

\[ \lambda x. \text{from}(x, i) \]

\[ \text{PP} \]

\[ \text{NP} \]

\[ \text{from} \]

\[ \text{np.i} \]

\[ \text{Author} \]

\[ \text{from} \]

\[ \text{India} \]
Florian vs Sandra: *the successful author from India*

```
D
  └── NP
      └── the

AP
  └── NP'
      └── λx[Author(x) ∧ from(x, i)]

    └── N
        └── Author(x)

    └── PP
        └── λx[from(x, i)]

        └── P
            └── N
                └── from
                    └── India

```

{ o.petukhova; n.dascalu }@lsv.uni-saarland.de
Florian vs Sandra: *the successful author from India*

\[
\lambda P \lambda x \forall P \forall x [\text{SuccessfulAs}(P)(x) \rightarrow P(x)]
\]

\[
\lambda x [\text{Author}(x) \land \text{from}(x, i)]
\]

\[
\lambda x. \text{Author}(x)
\]

\[
\lambda x. \text{from}(x, y)
\]

\[
\lambda y \lambda x \text{from}(x, y)
\]

\[
\lambda x [\text{from}(x, i)]
\]

\[
\lambda x [\text{Author}(x) \land \text{from}(x, i)]
\]

\[
\lambda P \forall x [\text{SuccessfulAs}(P)(x) \rightarrow P(x)]
\]

\[
\lambda x [\text{Author}(x) \land \text{from}(x, i)]
\]

\[
\lambda x. \text{Author}(x)
\]

\[
\lambda x. \text{from}(x, y)
\]

\[
\lambda y \lambda x \text{from}(x, y)
\]

\[
\lambda x [\text{from}(x, i)]
\]

\[
\lambda x [\text{Author}(x) \land \text{from}(x, i)]
\]
Florian vs Sandra: the successful author from India

a. DP

\[\text{the}\]

\[\lambda x \forall P \forall x [\text{SuccessfulAs} (\text{Author}(x) \land \text{from}(x, i)) \rightarrow (\text{Author}(x) \land \text{from}(x, i))]\]

\[\text{successful}\]

\[\lambda x [\text{Author}(x) \land \text{from}(x, i)]\]

\[\lambda x [\text{from}(x, i)]\]

\[\lambda y \lambda x [\text{from}(x, y)]\]
Florian vs Sandra: the successful author from India

\[ \lambda P. [\forall x. P(x)] \lambda x. \forall y. x [\text{SuccessfulAs}(\text{Author}(x) \land \text{from}(x, i)) \rightarrow (\text{Author}(x) \land \text{from}(x, i))] \]

\[ \lambda x. [\text{Author}(x) \land \text{from}(x, i)] \]

\[ \lambda x. \text{Author}(x) \]

\[ \lambda [\text{from}(x, y)] \]

\[ \lambda y. \lambda x. [\text{from}(x, y)] \]

\[ \text{successful} \]

\[ \text{the} \]

\[ \text{from} \]

\[ \text{India} \]

\[ \text{o.petukhova; n.dascalu}@lsv.uni-saarland.de \]
Florian vs Sandra: the successful author from India

a.
\[\lambda x.\forall P[\text{SuccessfulAs}(Author(x) \land from(x, i)) \to (Author(x) \land from(x, i))]\]
Florian vs Sandra: *the successful author from India*
Florian vs Sandra: the successful author from India
Florian vs Sandra: *the successful author from India*

\[
\text{DP} \\
\text{D} \quad \text{NP'} \\
\quad \text{the} \\
\quad \text{NP} \quad \text{PP} \\
\quad \text{AP} \quad \text{N} \\
\quad \quad \text{successful} \quad \text{author} \\
\quad \quad \text{P} \quad \text{N} \\
\quad \quad \quad \langle e, (e, t) \rangle \quad \langle e \rangle \\
\quad \quad \quad \lambda y \lambda x [\text{from}(x, y)] \quad \lambda i \\
\quad \quad \quad \quad \text{from} \quad \text{India}
\]
Florian vs Sandra: the successful author from India

b.

\[
\begin{align*}
\text{DP} & \quad \text{NP'} \\
\text{D} & \quad \text{NP} \\
\text{the} & \quad \text{PP} \langle e, t \rangle \\
\text{NP} & \quad \text{AP} \\
\text{successful} & \quad \text{N} \langle e, (e, t) \rangle \\
\text{author} & \quad \text{N} \\
\text{from} & \quad \text{from} \\
\text{India} & \quad \text{India}
\end{align*}
\]
Florian vs Sandra: *the successful author from India*

\[\lambda x[\text{from}(x, i)]\]

\[\lambda y \lambda x[\text{from}(x, y)]\]
Florian vs Sandra: *the successful author from India*

### b.

- **DP**
  - **D**
  - **NP’**
    - the
    - **NP**
      - **AP**
        - \(\langle e, t \rangle \langle e, t \rangle\)
      - \(\lambda P \lambda x \forall x [\text{SuccessfulAs}(P)(x) \rightarrow P(x)]\)
      - **successful**
      - **N**
        - \(\langle e, t \rangle\)
        - \(\lambda x. \text{Author}(x)\)
        - **author**
    - **PP**
      - \(\langle e, t \rangle\)
      - \(\lambda x [\text{from}(x, i)]\)
      - **from**
      - \(\langle e, (e, t) \rangle\)
      - **e**
      - \(\lambda y \lambda x [\text{from}(x, y)]\)
      - **from**
      - \(\langle e, i \rangle\)
      - **i**
      - **N**
        - \(\langle e, i \rangle\)
        - **india**
Florian vs Sandra: *the successful author from India*

```
b. the
    DP
        D
        NP'
        the
          NP
            ⟨⟨e, t⟩⟩
            λx.∀P∀x[SuccessfulAs(Author)(x) → Author(x)]
              AP
                ⟨⟨e, t⟩⟩
                λPλx∀P∀x[SuccessfulAs(P)(x) → P(x)]
                  N
                    ⟨e, t⟩
                    λx. Author(x)]
              N
                ⟨e, t⟩
                λx[from(x, i)]
                  PP
                    ⟨⟨e, t⟩⟩
                    λyλx[from(x, y)]
                      N
                        e
                        i
                      from
                        N
                          i
                          from
                          India
```
Florian vs Sandra: *the successful author from India*

b. DP

\[
\begin{align*}
\lambda x \forall y x \left[ \text{SuccessfulAs}(\text{Author})(x) \rightarrow \text{Author}(x) \land \text{from}(x, i) \right] \\
\lambda x \forall x \left[ \text{SuccessfulAs}(\text{Author})(x) \rightarrow \text{Author}(x) \right] \\
\lambda x \left[ \text{from}(x, i) \right] \\
\lambda P \forall y x \left[ \text{SuccessfulAs}(P)(x) \rightarrow P(x) \right] \\
\lambda x. \text{Author}(x) \\
\lambda y \lambda x \left[ \text{from}(x, y) \right] \\
\lambda i \\
\text{successful} \\
\text{author} \\
\text{from} \\
\text{India}
\end{align*}
\]
Florian vs Sandra: *the successful author from India*

b. DP

\[\lambda x \forall P \forall x [\text{SuccessfulAs}(\text{Author})(x) \rightarrow \text{Author}(x) \land \text{from}(x, i)]\]

[Diagram of the semantic representation of "the successful author from India" with formal expressions and symbols.]
Florian vs Sandra: the successful author from India

b. DP
   \[e\]
   \[
   \lambda x. \forall P [\text{SuccessfulAs}(\text{Author})(x) \rightarrow \text{Author}(x) \land \text{from}(x, i)]
   \]

D
\[
\langle \langle e, t \rangle, e \rangle
\]
\[
\lambda P. [\lambda x. P(x)]
\]
the

NP'
\[
\langle e, t \rangle
\]
\[
\lambda x. \forall P \forall x [\text{SuccessfulAs}(\text{Author})(x) \rightarrow \text{Author}(x) \land \text{from}(x, i)]
\]

NP
\[
\langle e, t \rangle
\]
\[
\lambda x. \forall P \forall x [\text{SuccessfulAs}(\text{Author})(x) \rightarrow \text{Author}(x)]
\]

AP
\[
\langle \langle e, t \rangle \langle e, t \rangle \rangle
\]
\[
\lambda P \lambda x. \forall P \forall x [\text{SuccessfulAs}(P)(x) \rightarrow P(x)]
\]
successful

N
\[
\langle e, t \rangle
\]
\[
\lambda x. \text{Author}(x)
\]
author

PP
\[
\langle e, t \rangle
\]
\[
\lambda y. \lambda x [\text{from}(x, y)]
\]
\[
i
\]
from

N
\[
\langle e, t \rangle
\]
\[
\lambda x. \text{india}(x)
\]
india
Topic 6: Beyond Function Application (Relative Clauses)
Topic 6: Beyond Function Application (Quantifier Raising)

Right Translation

everybody $\leadsto \lambda P\forall x.P(x)$
loves $\leadsto \forall x.Loves(m, x)$

$[\lambda P\forall x.P(x)][\lambda x.Loves(m, x)] \equiv \forall x.Loves(m, x)$

Solution: QUANTIFIER RAISING: syntactic transformation that moves a quantifier, an expression of type $\langle\langle e, t\rangle, t\rangle$, to a position in the tree where it can be interpreted, and leaves a DP trace in its previous position.
∀x. Loves(m, x)

\[ \lambda y. \forall x. \text{Loves}(y, x) \]

\[ \langle\langle e, t\rangle, \langle e, t\rangle\rangle \]

\[ \lambda Q(e, t) \]

\[ \forall x. \text{Loves}(y, x) \]

\[ \Rightarrow \text{RAISE-O} \]

\[ \langle e, \langle e, t\rangle\rangle \]

\[ \forall x. \text{Loves}(y, x) \]

\[ \lambda P. \forall x. P(x) \]

\[ \text{everybody} \]
Topic 6: Beyond Function Application (Two Quantifiers)

\[
S \\
\top \\
\exists y. \forall x. Loves(y, x)
\]

DP

\[
\langle \langle e, t \rangle, t \rangle \\
\lambda P. \exists y. P(y)
\]

| Somebody

VP

\[
\langle e, t \rangle \\
\lambda y. \forall x. Loves(y, x)
\]

\[
\langle \langle \langle e, t \rangle, \langle e, t \rangle \rangle, \langle e, t \rangle \rangle \\
\lambda Q_{(e,t),t} \lambda y. Q(\lambda x. Loves(y, x)) \\
\uparrow \text{RAISE-O} \\
\langle e, \langle e, t \rangle \rangle \\
\lambda x. \lambda y. Loves(y, x)
\]

| loves

NP

\[
\langle \langle e, t \rangle, t \rangle \\
\lambda P. \forall x. P(x)
\]

| everybody
Topic 6: Beyond Function Application (Type Shifting: RAISE-S)

\[
S(t) = \forall x \exists y. \text{Loves}(y, x)
\]

\[
\text{DP} \langle \langle e, t \rangle, t \rangle \quad \lambda P. \exists y. P(y)
\]

Somebody

\[
\text{VP} \langle \langle \langle e, t \rangle, t \rangle, t \rangle \quad \lambda Q_{(e,t),t}. \forall x. Q(\lambda y. \text{Loves}(y, x))
\]

\[
\text{V} \langle \langle \langle e, t \rangle, t \rangle, \langle \langle e, t \rangle, t \rangle, t \rangle \quad \lambda Q'_{(e,t),t}. Q'(\lambda x. Q(\lambda y. \text{Loves}(y, x)))
\]

\[
\uparrow_{\text{RAISE-O}} \langle e, \langle \langle e, t \rangle, t \rangle, t \rangle \quad \lambda x. \lambda Q_{(e,t),t}. Q(\lambda y. \text{Loves}(y, x))
\]

\[
\uparrow_{\text{RAISE-S}} \langle e, \langle e, t \rangle \rangle \quad \lambda x \lambda y. \text{Loves}(y, x)
\]

\[
\text{everybody}
\]

\[
\text{loves}
\]

---

Introduction into Formal Semantics, Summer 2021

{o.petukhova; n.dascalu}@lsv.uni-saarland.de
There are certain words/constructions that signal presupposition.

**Example**

Every man kissed the woman who loved him. >> There is at least one man and he kissed the woman who loved him.

Neither theory can explain why sentences whose presuppositions are not satisfied are odd. >> There are exactly two theories. (*quantifiers*)

John’s daughter will come. >> John has a daughter. (*possessives*)

John knows that Mary hates Bill. >> Mary hates Bill (*cognitive factive verbs*).

John is happy that Mary agrees to marry him. >> Mary agrees to marry John. (*emotive factive verbs*)

Mary bakes cookies again. >> Mary has baked cookies before. (*additive adverbs*)

It was John who broke the computer. >> Someone broke the computer. (*Clefts*)

The student is smart. >> There is an unique student in the context. (*definite determiners*)
Definite determiner ‘the’ is one of the presupposition triggers, what about others? Both, neither, every, etc.

∂ operator is defined like \( \partial(|P| = 2) \) reads ‘presupposing that there are exactly two Ps’.

\[
\text{neither} \leadsto \lambda P \lambda Q. [\partial(|P| = 2) \land \neg \exists x. [P(x) \land Q(x)]]
\]

\[
\text{every} \leadsto \lambda P \lambda Q. [\partial(\exists x. P(x)) \land \forall x. [P(x) \rightarrow Q(x)]]
\]
Neither candidate is qualified.

\[
S \quad \text{s}\partial (|\text{Candidate}| = 2 \land \neg \exists x. [\text{Candidate}(x) \land \text{Qualified}(x)])
\]

\[
\text{DP} \quad \lambda Q. [\partial (|\text{Candidate}| = 2 \land \neg \exists x. [\text{Candidate}(x) \land \text{Qualified}(x)])]
\]

\[
\text{VP} \quad \lambda x. \text{Qualified}(x)
\]

\[
\text{NP} \quad \lambda x. \text{Candidate}(x)
\]

\[
\land P \lambda Q. [\partial (|\text{Candidate}| = 2 \land \neg \exists x. [\text{Candidate}(x) \land \text{Qualified}(x)])]
\]

\[
\langle e, t \rangle
\]

\[
\land P \lambda Q. [\partial (|\text{Candidate}| = 2 \land \neg \exists x. [\text{Candidate}(x) \land \text{Qualified}(x)])]
\]

\[
\langle e, t \rangle
\]

\[
\text{DP} \quad e
\]

\[
\text{VP} \quad \langle e, t \rangle
\]

\[
\text{NP} \quad \langle e, t \rangle
\]

\[
\text{DP} \quad \langle e, t \rangle
\]

\[
\text{VP} \quad \langle e, t \rangle
\]

\[
\text{NP} \quad \langle e, t \rangle
\]

\[
\text{DP} \quad \langle e, t \rangle
\]

\[
\text{VP} \quad \langle e, t \rangle
\]

\[
\text{NP} \quad \langle e, t \rangle
\]

\[
\text{DP} \quad \langle e, t \rangle
\]

\[
\text{VP} \quad \langle e, t \rangle
\]

\[
\text{NP} \quad \langle e, t \rangle
\]

\[
\text{DP} \quad \langle e, t \rangle
\]

\[
\text{VP} \quad \langle e, t \rangle
\]

\[
\text{NP} \quad \langle e, t \rangle
\]
Topic 7: Presupposition (Weak vs Strong Kleene Connectives)

**Weak Star Idea:** We see as a # “contaminating” (or nonsense) value, which does not allow us to deduce anything if there is a presupposition failure somewhere.

**Strong Star Idea:** Idea: We see # in one argument as “ignorance” (unknown) - it still allows us to deduce the result from the value of the other argument.
LaPierre (1992) defines identity between two terms as follows:

- if neither $\alpha$ nor $\beta$ denotes the undefined individual, then $\llbracket \alpha = \beta \rrbracket^{M,g} = 1$ if $\llbracket \alpha \rrbracket^{M,g} = \llbracket \beta \rrbracket^{M,g}$, and 0 otherwise.
- If one $\alpha$ or $\beta$ denotes the undefined individual, then $\llbracket \alpha = \beta \rrbracket^{M,g} = 0$
- If both denote the undefined individual, then $\llbracket \alpha = \beta \rrbracket^{M,g} = \#$ is undefined (not enough is “known” about the objects to determine that they are the same or distinct).
Topic 7: Presupposition (Predication with Undefined Individuals)

Semantic Rule: Existence Predicate

$[[\text{Exists}(\alpha)]_{M,g} = 1 \text{ if } [\alpha]_{M,g} \neq \#_e \text{ and } 0 \text{ otherwise}$

Predict truth value of ‘The Golden Mountain does not exist’
Topic 8:
A situation, process, action etc. or the verb phrase, sentence, etc, expressing this situation, etc, has the property iff

- It is directed toward attaining a goal or limit at which the action exhausts itself and passed into something else (Andersson, 1972)
- It leads up to a well-defined point behind which the process cannot continue (Comrie, 1976)
- has actual or potential terminal point (Dahl, 1981)

A terminal point $t$ is defined such that

1. if $t$ is reached, the process cannot continue
2. $t$ will be reached in the normal course of events (≡ if nothing unexpected intervenes)
3. $t$ will be reached in all possible courses of events.
States are static, extended in time, and lack a natural end point.

Activities are like states except they typically involve or lead to some kind of change.

Accomplishments are like activities except they have a natural end point.

Achievements are like accomplishments except they are punctual rather than extended in time.

<table>
<thead>
<tr>
<th>State</th>
<th>( V(x_1, \ldots, x_n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity</td>
<td>( DO(x_1, V(x_1, \ldots, x_n)) )</td>
</tr>
<tr>
<td>Accomplishment</td>
<td>( DO(x_1, V(x_1, \ldots, x_n)) CAUSE(BECOME \ V(x_1, \ldots, x_n)) )</td>
</tr>
<tr>
<td>Achievement</td>
<td>( (BECOME \ V(x_1, \ldots, x_n)) )</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. Mary walked miles</td>
</tr>
<tr>
<td>V(Act)</td>
</tr>
<tr>
<td>b. Mary walked three miles</td>
</tr>
<tr>
<td>V(Acc)(↑ V(Act))</td>
</tr>
<tr>
<td>c. My program ran for a few minutes.</td>
</tr>
<tr>
<td>V(Act)</td>
</tr>
<tr>
<td>d. My program ran in less than four minutes (this morning).</td>
</tr>
<tr>
<td>V(Acc)(↑ V(Act))</td>
</tr>
<tr>
<td>e. Suddenly, I knew the answer.</td>
</tr>
<tr>
<td>V(Ach)(↑ V(State))</td>
</tr>
<tr>
<td>f. John played the sonata for about eight hours.</td>
</tr>
<tr>
<td>V(Acc)(↑ V(Act))</td>
</tr>
<tr>
<td>g. For months, the train arrived late.</td>
</tr>
<tr>
<td>V(State)(↑ V(Ach))</td>
</tr>
</tbody>
</table>
### Topic 9: Aspect & Tense (inner aspectuality)

**Example**

(a) Mary walked three miles
\[ V[+Addto] + NP(int)[+SQA] \sim [+T(VP)] \]

(b) Mary walked miles
\[ V[+Addto] + NP(int)[-SQA] \sim [-T(VP)] \]

Calculate the plus/minus value of terminativity at the sentence level:

**Example**

a. [S Mary [VP walk [NP three miles]]]
\[ [+Ts[+SQA]] [+Tv[p][+ADDTO]] [+SQA]] (terminative)

b. [S Mary [VP walk [NP miles]]]
\[ [-Tts[+SQA]] [-Tv[p][+ADDTO]] [-SQA]] (durative)

c. [S Children [VP walk [NP three miles]]]
\[ [-Tts[−SQA]] [+Tv[p][+ADDTO]] [+SQA]] (durative)

d. [S Mary [VP save [NP three miles]]]
\[ [-Tts[+SQA]] [-Tv[p][−ADDTO]] [+SQA]] (durative)
## Topic 9: Aspect & Tense (Viewpoint Aspect)

<table>
<thead>
<tr>
<th>Temporal relations</th>
<th>Tense category</th>
<th>Traditional label</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. E–R–S</td>
<td>Anterior Past</td>
<td>Past Perfect</td>
<td>I had passed the exam by the end of the winter.</td>
</tr>
<tr>
<td>2. E,R–S</td>
<td>Simple Past</td>
<td>Simple Past</td>
<td>I passed the exam.</td>
</tr>
<tr>
<td>3.a R–E–S</td>
<td>Posterior Past</td>
<td>–</td>
<td>I did not know that he would win. (yesterday)</td>
</tr>
<tr>
<td>3.b R–S,E</td>
<td>Posterior Past</td>
<td>–</td>
<td>I did not know that he would be here. (right now)</td>
</tr>
<tr>
<td>3.c R–S–E</td>
<td>Posterior Past</td>
<td>–</td>
<td>I did not know that he would come. (tomorrow)</td>
</tr>
<tr>
<td>4. E–S,R</td>
<td>Anterior Present</td>
<td>Present Perfect</td>
<td>I have passed the exam.</td>
</tr>
<tr>
<td>7a S–E–R</td>
<td>Anterior Future</td>
<td>Future Perfect</td>
<td>I will have passed the exam by the end of the winter.</td>
</tr>
<tr>
<td>7.b S,E–R</td>
<td>Anterior Future</td>
<td>Future Perfect</td>
<td>John will have fixed the car by tonight. (already repaired)</td>
</tr>
<tr>
<td>7.c E–S–R</td>
<td>Anterior Future</td>
<td>Future Perfect</td>
<td>I will have fixed the car by tonight. (just repairing it)</td>
</tr>
</tbody>
</table>
[Tense[Aspect * [eventuality description]] where the Kleene star * indicates zero, one or more operations.

\[
\begin{align*}
\text{TenseP} & \quad \text{Tense} \quad \text{AspP} \\
& \quad \text{Asp} \quad \ldots
\end{align*}
\]

**PERF** \( \leadsto \lambda P_{\langle i,t \rangle}. \lambda t. \exists t'. t' \subseteq t \land P(t') \)

**IMP** \( \leadsto \lambda P_{\langle i,t \rangle}. \lambda t. \exists t'. t \subseteq t' \land P(t') \)

**PAST** \( \leadsto \iota t. [t = t_n \land t_n < \text{now}] \)

**PRESENT** \( \leadsto \text{now} \)

**WOLL** \( \lambda P_{\langle i,t \rangle}. \lambda t. \exists t'. t < t' \land P(t') \)
Topic 9: Aspect & Tense (Derivation for past (imperfective))

\[ \exists t'.[\lambda t. [t = t_n \land t_n < \text{now}]] \subseteq t' \land \text{Dance}(t', a) \]

\[ \lambda t. \exists t'. t \subseteq t' \land \text{Dance}(t', a) \]

\[ \lambda P_{(i,t)} \lambda t. \exists t'. t \subseteq t' \land P(t') \]

\[ \text{Anna dance} \]
Example 2

(1) Brutus was a famous Roman politician.
(2) Brutus was a Roman politician.
(3) Brutus was a famous politician.
(4) Brutus was a politician.

Example 2

(1) Brutus stabbed Caesar at noon on the forum.
(2) Brutus stabbed Caesar on the forum.
(3) Brutus stabbed Caesar at noon.
(4) Brutus stabbed Caesar.
Event Semantics: Advantages of the Neo-Davidsonian approach

- Makes it easier to state generalizations across the categories of nouns and verbs, and to place constraints on thematic roles
- Good for formulating analyses without committing to an argument/adjunct distinction
- Lends itself to a natural compositional process in terms of intersection with an existential quantifier at the end

Example

a. \([[\text{agent}]]) = \lambda x \lambda e[\text{agent}(e) = x]

b. \([[\text{theme}]]) = \lambda y \lambda e[\text{agent}(e) = y]

c. \([\text{stab}] = \lambda e[\text{stab}(e)]

d. \([[\text{agent}][\text{Brutus}]) = \lambda e[\text{agent}(e) = \text{brutus}]

e. \([[\text{theme}][\text{Caesar}]) = \lambda e[\text{agent}(e) = \text{caesar}]

f. \([\text{Brutus stab Caesar }] = (c) \cap (d) \cap (e)

g. \([\text{Brutus stab Caesar }] = \exists e \in (c) \cap (d) \cap (e)

(sentence radical)

(full sentence)
**Topic 10: Event Semantics (composition)**

### Predicates

- $\text{stab} \sim \lambda e.\text{Stab}(e)$
- $\text{butter} \sim \lambda e.\text{Butter}(e)$

### Syntax

$\text{DP} \rightarrow \theta \text{ DP}$

### Lexicon

$\theta$: [agent], [theme], [instrument], [recipient], [goal], [location], [time], ...

### theta Mapping

- $[\text{agent}] \sim \lambda x\lambda e.\text{agent}(e) = x$
- $[\text{theme}] \sim \lambda x\lambda e.\text{theme}(e) = x$
- $[\text{instrument}] \sim \lambda x\lambda e.\text{instrument}(e) = x$
- ...

{o.petukhova; n.dascalu}@lsv.uni-saarland.de

Introduction into Formal Semantics, Summer 2021
Topic 10: Event Semantics (existential closure)

\[ S \]

\[ t \]

\[ \exists e. Barks(e) \land agent(e) = s \]

\[ \uparrow \]

\[ \langle v, t \rangle \]

\[ \lambda e. Bark(e) \land agent(e) = s \]

\[ \text{DP} \]

\[ \langle v, t \rangle \]

\[ \lambda e. agent(e) = s \]

\[ \lambda e. Barks(e) \]

\[ \text{VP} \]

\[ \theta \]

\[ \langle e, \langle v, t \rangle \rangle \]

\[ \lambda x \lambda e. agent(e) = x \]

\[ [\text{agent}] \]

\[ \text{barks} \]

\[ \text{Spot} \]

\[ \lambda x \lambda e. agent(e) = x \]

\[ s \]

\[ \text{Spot} \]

\[ \{\text{o.petukhova; n.dascalu}@lsv.uni-saarland.de} \]

Introduction into Formal Semantics, Summer 2021
Topic 10: Event Semantics (verbs as event quantifiers)

a. \( \lambda f \exists e. [Barks(e) \land agent(e) = s \land f(e)](\lambda e. true) \)
b. \( \exists e. [Barks(e) \land agent(e) = s \land (\lambda e. true)(e)] \)
c. \( \exists e. [Barks(e) \land agent(e) = s \land true] \)
d. \( \exists e. [Barks(e) \land agent(e) = s] \)

Type-Shifting Rule 6: Quantifier Closure

if \( \alpha \leadsto \alpha' \), where \( \alpha' \) is of a category \( \langle\langle v, t\rangle, t\rangle \), then:
\[ \alpha \leadsto \alpha'(\lambda e. true) \]
as well.


**Topic 10: Event Semantics (verbs as event quantifiers)**

\[
S \\
t \\
\exists e. Barks(e) \land agent(e) = s \\
\uparrow \\
\langle \langle v, t \rangle, t \rangle \\
\lambda f \exists e. Barks(e) \land agent(e) = s \land f(e)
\]

\[
\text{DP} \\
\langle \langle \langle v, t \rangle, t \rangle, \langle \langle v, t \rangle, t \rangle \rangle \\
\lambda V \lambda f. V(\lambda e. agent(e) = s \land f(e)) \\
\theta \\
\langle e, \langle \langle v, t \rangle, t \rangle, \langle \langle v, t \rangle, t \rangle \rangle \\
\lambda x \lambda V \lambda f. V(\lambda e. agent(e) = x \land f(e)) \\
e \\
\lambda x V \lambda f. V(\lambda e. agent(e) = x \land f(e)) \\
s \\
[\text{agent}] \\
\text{Spot}
\]

barks

\[\{o.petukhova; n.dascalu\}@lsv.uni-saarland.de\]
Topic 10: Event Semantics (other quantifiers)

- Quantificational Noun Phrase
- Quantificational Adjuncts
- Negation and its scope