Semantic Theory
Lecture 12: Events and Processes; Semantic Roles

Manfred Pinkal

FR 4.7 Computational Linguistics and Phonetics

Summer 2014

Final Exam

Thursday, July 24

10:00 -

Seminar Room

Event Semantics

- A model structure with events and temporal precedence is defined as $\mathbf{M = \langle U, E , <, e_u, V \rangle}$, where
  - $U \cap E = \emptyset$,
  - $< \subseteq E \times E$ is a partial ordering relation (temporal precedence)
  - $e_u \in E$ is the utterance event
  - V is an interpretation function like in standard FOL, with $D_e = U \cup E$.

Model structures for plural terms

- A model structure is a pair **M = ⟨⟨U, ≤⟩, V⟩**, where
  - ⟨U, ≤⟩ is an atomic join semi-lattice with universe U and individual part relation ≤.
  - V is a value assignment function.
- A ⊆ U is the set of atoms in ⟨U, ≤⟩.
- U \ A is the set of non-atomic elements, i.e., the proper sums or groups in U.

Model Structure for Mass Terms

- We add another sort of entities, the **“portions of matter”** M, to the model structure, and distinguish an individual part and a material part relation, writing $\leq_i$ for the former, and $\leq_m$ for the latter:

- **$\mathbf{M} = \langle\langle U, \leq_i\rangle, \langle M, \leq_m\rangle, h, V\rangle$**
  - $U \cap M = \emptyset$
  - $\langle U, \leq_i\rangle$ is an atomic join semi-lattice
  - **$\langle M, \leq_m\rangle$ is a non-atomic** (and dense) **join semi-lattice**
  - V is a value assignment function

Vendler‘s Aspectual Verb Classes

States

know, believe, own, love, resemble

Eventualities

Activities

run, walk, swim, work, sleep, rain

Events

Accomplishments

paint a picture, write a paper, build a house

Achievements

recognize, spot, find, lose, reach, die

UNIVERSITÄT DES SAARLANDES

Model Structure with Sub-Events

- In analogy to plural semantics, we can represent sub-event relations by a join semi-lattice.
- $M = \langle U, \langle E, \leq_e \rangle , <, e_u, V \rangle$, where
  - $U \cap E = \emptyset$,
  - $< \subseteq E \times E$ is a partial ordering relation (temporal precedence)
  - $e_u \in E$ is the utterance event
  - **$\langle E, \leq_e \rangle$ is a join semi-lattice**
  - V is an interpretation function

Model Structure with Sub-Events

- $M = \langle U, \langle E, \leq_e \rangle , <, e_u, V \rangle$, where
  - $U \cap E = \emptyset$,
  - $< \subseteq E \times E$ is a partial ordering relation (temporal precedence)
  - $e_u \in E$ is the utterance event
  - $\langle E, \leq_e \rangle$ is a join semi-lattice
  - V is an interpretation function
- The model structure must observe some additional constraints on $<$ and $\leq_e$, e.g.:
  - If $e_1 < e_2$ and $e_1' \leq_e e_1$ and $e_2' \leq_e e_2$, then $e_1' < e_2'$
  - If $e_1' \circ e_2'$ and $e_1' \leq_e e_1$ and $e_2' \leq_e e_2$, then $e_1 \circ e_2$

Model Structure with Sub-Events

- **Application:** Complex events are represented as sequences of temporally ordered sub-events
  - for instance “scripts” like: *visit a restaurant* or *shopping in the supermarket*

Processes and Mass Terms

- Process-describing verbs are similar to mass terms. Both are
  - Cumulative:

    gold(x), gold(y) $\models$ gold($x \oplus_m y$)

    rain(e_1), rain(e_2) $\models$ rain($e_1 \oplus_e e_2$)
  - Divisive:

    gold(x), $y \lhd_m x$, $\models$ gold(y)

    rain(e_1), $e_2 \lhd_e e_1$, $\models$ rain(e_2)

Processes and Mass Terms

- In analogy to the semantics of mass terms, assume
  - a domain of processes ("event matter") in addition to the domain of individual events, represented through a non-atomic join semi-lattice
  - a "materialisation function" for events that maps individual events to processes

    $M = \langle\langle U, \leq_i\rangle, \langle M, \leq_m\rangle, h, \langle E_i, \leq_{ei}\rangle, \boldsymbol{\langle E_m, \leq_{em}\rangle}, \boldsymbol{h_e}, <, e_u, V\rangle$

- Add two-place relations $\boldsymbol{\lhd_{ei}}$, $\boldsymbol{\lhd_{em}}$, and operators $\boldsymbol{\oplus_{ei}}$, $\boldsymbol{\oplus_{em}}$, and a function expression m_e to the representation language, and give them a straightforward semantic interpretation in terms of $\boldsymbol{\leq_{ei}}$, $\boldsymbol{\leq_{em}}$, $\boldsymbol{\sqcup_{ei}}$, $\boldsymbol{\sqcup_{em}}$, $\boldsymbol{h_e}$.

Progressive Form

(1) *John is eating an apple*

- The core of the interpretation of progressive form is the materialization function h_e, which maps individual events – the telic action of John’s eating an apple – to the process or activity that leads to the result.

(Very Preliminary) Interpretation of the Progressive Form

(1) John is eating an apple

- **Progressive operator:**
  - PROG := $\lambda E \lambda e' \exists e[E(e) \wedge e'=m_e(e)]$
  - $\lambda E \lambda e' \exists e[E(e) \wedge e'=m_e(e)](\lambda e'' \exists x[apple(x) \wedge eat(e'',j^*,x)])$
  - $\Leftrightarrow_\beta \lambda e' \exists e[\exists x[apple(x) \wedge eat(e,j^*,x)] \wedge e'=m_e(e)]$

- **Present progressive:**
  - PRES := $\lambda E\ \exists e''[E(e'') \wedge e'' \circ e_u]$
  - $\lambda E\ \exists e''[E(e'') \wedge e'' \circ e_u]$
    $(\lambda e' \exists e[\exists x[apple(x) \wedge eat(e,j^*,x)] \wedge e'=m_e(e)])$
  - $\Leftrightarrow_\beta \exists e''[\exists e \exists x[apple(x) \wedge eat(e,j^*,x)] \wedge e''=m_e(e) \wedge e'' \circ e_u]$

Semantic Roles: An Example

(1) *The window broke*

(2) *A rock broke the window*

(3) *John broke the window with a rock*

(1) *[John]$_{ag}$ broke [the window]$_{pat}$ [with a rock]$_{inst}$*

(2) *[A rock]$_{inst}$ broke [the window]$_{pat}$*

(3) *[The window]$_{pat}$ broke*

A Variant of Davidson‘s Problem?

- How do we model entailment?

$$\text{break}_3(\mathbf{j}, \mathbf{w}, \mathbf{r}) \vDash \text{break}_2(\mathbf{r}, \mathbf{w}) \vDash \text{break}_1(\mathbf{w})$$

- This reminds of Davidson‘s problem:

$$\text{kill}_4(g, b, m, p) \Rightarrow \text{kill}_3(g, b, p) \Rightarrow \text{kill}_2(g, b)$$

- A solution along the lines of Davidson’s event semantics:
  - Introduce an event argument
  - Represent roles as binary relations between events and participants

„Neo-Devidsonian“ Event Semantics

- Assume an implicit event argument for event verbs (we need it anyway).
- Represent roles as binary relations between events and participants:

(1) $\exists e\ [break(e) \wedge pat(e, w)]$

(2) $\exists e\ [break(e) \wedge pat(e, w) \wedge inst\ (e, r)]$

(3) $\exists e\ [break(e) \wedge ag(e, j) \wedge pat(e, w) \wedge inst\ (e, r)]$

Differences

- Event modifiers are
  - syntactically realize by free adjuncts
  - freely applicable to all event verbs, and
  - can be iteratively applied to event predicates in arbitrary number
- Semantic roles are
  - syntactically realized by complements,
  - which can occur with a verb only in accordance with verb-specific subcategorization constraints

Differences

- Adjuncts expressing event modifiers are semantically transparent (modulo ambiguity): the adjunct *at midnight* expresses a temporal modifier, *in the park* a location.
- Syntactic complements realize different semantic roles, and one role can be realized by different complement types. The relation between roles and their syntactic realizations ("**role-linking relation**") is verb-specific.
- Adjuncts express "external" properties of events.
- Semantic roles refer to intrinsic parts of the event structure.

What are Semantic Roles?

- Understanding a verb (or any other predicate) means to know the situation type or **conceptual schema** (the “**frame**”) associated with or evoked by it.
- Part of the situation type or conceptual schema are typical **participants**: persons or objects that play a specific **role** in the conceptual schema expressed by the predicate.

How many Roles?

- According to Fillmore (1968), **roles are universal**: they form a small, closed inventory.
  - A typical role inventory: Agent, Theme (Patient, Object),Recipient, Instrument, Source, Goal, Beneficiary, Experiencer.

- *[Mary]$_{Ag}$ gave [a book]$_{Pat}$ [to John]$_{Rec}$*
- *[John]$_{Rec}$ received [a book]$_{Pat}$ [from Mary]$_{Ag}$*
- *But: [Mary]$_{???}$ sold [a car]$_{???}$ [to John]$_{???}$ [for 3,000 €]$_{???}$*

How many Roles?

- Specific role inventories for **each lemma**:

  roles of *kick*: $\text{arg0}_{\text{kick}}$, $\text{arg1}_{\text{kick}}$ or „kicker“, „kicked“

- This is the **PropBank** solution.
- Problem: Cross-lexical relations (and entailments) cannot be modelled:

  give : receive

  buy : sell

  like : please

How many Roles?

- Specific role inventories for different **frames**: Event or situation schemata that are „evoked“ by content words, typically verbs (also called frame-evoking elements or target words).
- Semantic roles are neither universal nor lemma-specific: There are typically several target words for a frame. Roles apply across the target words of a frame.
- This is the **FrameNet** variant of role semantics.
- Example: The “Commercial Transaction” frame is evoked by *sell, buy, vend, auction, purchase, sale, …* and has frame-specific roles (“frame elements”) Seller, Buyer, Goods, Money.

Roles in Compositional Semantics

- How do we get from a surface sentence to its role-semantic representation?
- Two sub-tasks:
  - **Role Linking:** How can syntactic relations between verb and arguments be mapped to semantic roles?
  - **Semantic Construction:** How can we integrate role information in type theory?

Role Linking

- Part of the linking process is regular. For instance:
  - An overt agent always becomes subject.
  - If there is no overt agent, the instrument becomes subject.
  - If no agent or instrument occurs, the theme becomes subject.
- Linguistic grammar theories try to describe role linking systematically, as part of the grammar, working, e.g., with “obliqueness hierarchies”.
- Problem: Role linking has idiosyncratic aspects.
- As a consequence: Linking information should be (to some part) provided by the lexicon.
- (Statistical role labelers typically exploit grammatical as well as lexical regularities.)

Semantic Composition (just for illustration!)

- Order-free lambda abstraction

- *kick* ⇒ λ{x, y, e}.kick'[ref:e, ag:x, pat:y]
- *kick Bill* ⇒ λ{x, y, e}.kick'[ref:e, ag:x, pat:y](bill'$_{pat}$)

  ⇔ λ{x, e}.kick'[ref:e, ag:x, pat:bill']
- *Mary kicked Bill* ⇒ λ{x, e}.kick'[ref:e, ag:x, pat:bill'](mary'$_{ag}$)

  ⇔ λ{e}.kick'[ref:e, ag:mary', pat:bill']