Semantic Theory
Lecture 10: Event Semantics

Manfred Pinkal
FR 4.7 Computational Linguistics and Phonetics

Summer 2014

UNIVERSITÄT
DES
SAARLANDES

Verbs

- *John likes Mary*
- *Mary kicked John*
- *Bill is coughing*
- *Bill travelled to Paris*

Donald Davidson‘s Problem

(1) The gardener killed the baron at midnight in the park

$\Rightarrow kill_4(g, b, m, p)$

(2) The gardener killed the baron at midnight

$\Rightarrow kill_3(g, b, m)$

(3) The gardener killed the baron in the park

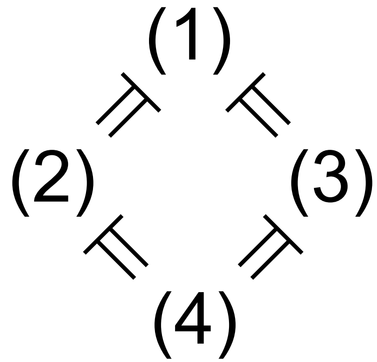
$\Rightarrow kill_2(g, b, p)$

(4) The gardener killed the baron

$\Rightarrow kill_1(g, b)$

Davidson's Problem

- How can the systematic entailment relations between the different uses of *kill* be explained?



- Naïve FOL interpretation does not answer this question:
  - $kill_4(g, b, m, p) \not\models kill_3(g, b, m)$
  - $kill_3(g, b, m) \not\models kill_1(g, b)$
  - etc.

A possible solution?

- Determine the maximum arity n of the predicate.
- Take n to be the arity of the predicate.
- Bind syntactically empty argument positions with existential quantification.

  (1) $\Rightarrow$ kill(g, b, m, p)

  (2) $\Rightarrow$ $\exists$y kill(g, b, m, y)

  (3) $\Rightarrow$ $\exists$x kill(g, b, x, p)

  (4) $\Rightarrow$ $\exists$x$\exists$y kill(g, b, x, y)
- But: What is the maximum arity of a predicate?

  The gardener killed the baron at midnight in the park under cover of absolute darkness with a gun out of jealousy …

Event Arguments

- Davidson's Solution: Verbs denote events.
- Example: The transitive *kill* is represented by a three-place relation kill'. The first argument of kill' is an event variable, which existentially bound:

  $\exists e\ kill'(e,g,b)$

- In general, n-place event verbs are represented by relations of arity n+1.
- Adjuncts denote two-place relations between an event and a time, a location, or other kinds of circumstantial information.

Davidson‘s Problem Solved

The gardener killed the baron at midnight in the park

$\Rightarrow \exists e\ [kill'(e,g,b) \wedge time(e,m) \wedge location(e,p)]$

- Entailment problem is solved:

$$\exists e\ [kill'(e,g,b) \wedge time(e,m) \wedge location(e,p)]$$

$$\models \qquad\qquad \models$$

$$\exists e\ [kill'(e,g,b) \wedge time(e,m)] \qquad \exists e\ [kill'(e,g,b) \wedge location(e,p)]$$

$$\models \qquad\qquad \models$$

$$\exists e\ [kill'(e,g,b)]$$

- Arbitrary number of adjuncts can be naturally represented

Event Verbs and Nominal Event Predicates

- *Bill saw an elephant.*

  $\exists e \exists x$ [see(e, b, x) ∧ elephant(x)]

- *Bill saw an accident.*

  $\exists e \exists e'$ [see(e, b, e') ∧ accident(e')]

- *Bill saw the children play*

  $\exists e \exists e'$ [see(e, b, e') ∧ play(e', the-children)]

Model Structure with Events

- We enrich model structures with ontological information in the traditional Aristotelian sense of ontology:
- Ontology is the area of philosophy identifying and describing the basic “categories of being and their relations”

Model Structure with Events

- We assume two disjoint classes, or kinds, or sorts of entities:
  - A set of “standard individuals” or “objects” U
  - A set of events E
- A model structure with events is a triple

  M = ⟨U, E, V⟩,

  with a set of standard individuals U,

  a set of events E, and

  an interpretation function V.
- Note: Both standard objects and events are both possible denotations of type e expressions. Entities (in the general sense) may be either standard objects or events.

Sorted Logic

- We a separate inventory of individual variables (Type e variables) for each of the two sorts of individuals in addition to the general, sort-unspecific variable set.
- **(Standard) Object variables:** Var_U = u, v, w, ...

  (or just: x, y, z, ...; see below)
- **Event variables:** Var_E = e, e', e'', ..., , e_1, e_2, ...
- **Note:** Both standard object and event variables are of type e. Formally, we have to distinguish two types of sorted variables plus general variables without sortal restrictions. Practically, we often collapse standard-object and general variables, and assume that the disambiguation becomes clear from context.

Sorted Logic

- A variable assignment function g assign variables an individual of the appropriate sort-specific domain:
  - $g(u) \in U$ for $u \in Var_U$
  - $g(e) \in E$ for $e \in Var_E$

- Quantification ranges over sort-specific domains:
  - $[\![\exists u\ \Phi\ ]\!]^{M,g} = 1$ iff there is an $a \in U$ s.t. $[\![\ \Phi\ ]\!]^{M,g[u/a]} = 1$
  - $[\![\exists e\ \Phi\ ]\!]^{M,g} = 1$ iff there is an $a \in U$ s.t. $[\![\ \Phi\ ]\!]^{M,g[e/a]} = 1$

Event-Denoting Nouns and Verbs

- Events as first-class ontological entities simultaleously solve several problems of semantic representation.
- We can model:
  - event-denoting nouns (*travel, accident, lecture*) and standard-object denoting nouns
  - verbs taking overt event arguments (*start, end, last*),
  - verbs which are unspecific w.r.to the argument sort (*see*)
  - events being alternatively realize by nouns and verbs (*travel*)

Sortal Constraints

- Non-logical constants of sortal constraints come with *sortal constraints* on their argument positions.
  - accident‘ takes an event argument
  - start‘ takes an implicit and an overt event argument
  - see‘ takes (1) an (implicit) event argument, (2) a standard-object argument, and (3) an argument that can be either event or standard object.

Event Semantics: Compositional Derivation of Adjuncts

- Treatment of adjuncts as predicate modifiers, in analogy to attributive adjectives: type ((e,t),(e,t)):
- Intersective adjectives modify nominal predicates:
  - Representation of the intersective adjective *red:*

    red $\Rightarrow \lambda F \lambda x[F(x) \wedge \text{red}^*(x)]$,

    modifying, e.g., $\lambda x[\text{book'}(x)]$
- Adjuncts modify event predicates, represented by sentences:
  - *at midnight* $\Rightarrow \lambda E \lambda e[E(e) \wedge \text{time}(e, \text{midnight})]$,

    modifying, e.g., $\lambda e[\text{kick'}(e, m^*, j^*)]$

Event Semantics: Compositional Derivation of Adjuncts

- *kill* ⇒ $\lambda y\, \lambda x\, \lambda e\, [\, kill(e, x, y)\,] : \langle e,\langle e,\langle e,t\rangle\rangle\rangle$
- *the baron* ⇒ $b : e$
- *the gardener* ⇒ $g : e$
- *at midnight* ⇒ $\lambda F \lambda e[F(e) \wedge time(e, midnight)] : \langle\langle e,t\rangle,\langle e,t\rangle\rangle$
- *in the park* ⇒ $\lambda F \lambda e[F(e) \wedge location(e, park)] : \langle\langle e,t\rangle,\langle e,t\rangle\rangle$

Event Semantics: Compositional Derivation of Adjuncts

- *The gardener killed the baron*

  $\Rightarrow \lambda y\ \lambda x\ \lambda e\ [\ kill(e, x, y)\ ](g)(b)$

- *... at midnight*

  $\Rightarrow \lambda F\ \lambda e\ [\ F(e) \wedge time(e, midnight)\ ](\lambda e\ [\ kill(e, g, b)\ ])$

  $\Leftrightarrow \lambda e\ \ [\ kill(e, g, b) \wedge time(e, midnight)\ ]$

- *... in the park*

  $\Rightarrow \lambda F \lambda e[F(e) \wedge location(e, park)](\lambda e[kill(e, g, b) \wedge time(e, m.)])$

  $\Leftrightarrow \lambda e[kill(e, g, b) \wedge time(e, midnight) \wedge location(e, park)]$

- *"Existential closure":*

  $\boldsymbol{\exists e}[kill(e, g, b) \wedge time(e, midnight) \wedge location(e, park)]$

Adjuncts and Modifiers [1]

- Uniform semantic representation for adjuncts and post-nominal modifiers:

  in the park $\Rightarrow$ $\lambda F \lambda x[F(x) \wedge \text{location}(x, \text{park})]$

- **Adjunct**:

  $[_S[_S$*The gardener killed the baron* $]$ $[_{PP}$*in the park*$]]$

  $\Rightarrow$ $\lambda F \lambda x[F(x) \wedge \text{location}(x, \text{park})](\lambda e.\text{kill}(e, g, b))$

  $\Leftrightarrow$ $\lambda e[\text{kill}(e, g, b) \wedge \text{location}(e, \text{park})]$

Adjuncts and Modifiers [2]

- Uniform semantic representation for adjuncts and post-nominal modifiers:

  in the park $\Rightarrow$ $\lambda F \lambda x[F(x) \wedge \text{location}(x, \text{park})]$

- **Post-nominal modifier of an event noun**:

  (a) $[_N [_N$ *felony*$]$ $[_{PP}$ *in the park*$]]$

  $\Rightarrow$ $\lambda F \lambda x[F(x) \wedge \text{location}(x, \text{park})](\lambda e.\text{felony}(e))$

  $\Leftrightarrow$ $\lambda e[\text{felony}(e) \wedge \text{location}(e, \text{park})]$

Adjuncts and Modifiers [2]

- Uniform semantic representation for adjuncts and post-nominal modifiers:

  in the park ⇒ $\lambda F \lambda x[F(x) \wedge \text{location}(x, \text{park})]$

- **Post-nominal modifier of a standard noun**:

  (a) $[_N\, [_N$ *fountain*$]\ [_{PP}$ *in the park*$]]$

  $\Rightarrow\ \lambda F \lambda x[F(x) \wedge \text{location}(x, \text{park})](\lambda e.\text{fountain}(e))$

  $\Leftrightarrow\ \lambda x[\text{fountain}(x) \wedge \text{location}(x, \text{park})]$

Tense

- Natural-language sentences are tensed:

  John is walking

  John walked

  John will walk

- Representation of tense in conventional tense logic:

  walk(john)

  P*walk(john)*

  F*walk(john)*

Classical Tense Logic

- Tense-logical model structure: M = <U, T, <, V>
  - U, T non-empty sets, $U \cap T = \emptyset$
  - < a linear ordering on T
  - V a value assignment function, which assigns to every non-logical constant α a function from T to appropriate denotations of α

- Interpretation of tense operators:

  $[\![\mathbf{P}A]\!]^{M,t} = 1$ iff $[\![A]\!]^{M,t'} = 1$ for at least one $t' < t$

  $[\![\mathbf{F}A]\!]^{M,t} = 1$ iff $[\![A]\!]^{M,t'} = 1$ for at least one $t' > t$

Temporal Relations in Natural Language

- *The door opened, and Mary entered the room.*
- *John arrived. Then Mary left.*
- *Mary left, before John arrived.*
- *John arrived. Mary had left already.*

Event Semantics: An Alternative Treatment of Tense

- A temporally ordered event structure is defined as

  $M = (U, E, <, e_u, V)$,

  with $U \cap E = \varnothing$,

  $< \subseteq E \times E$ an asymmetric relation (temporal precedence)

  $e_u \in E$ the utterance event

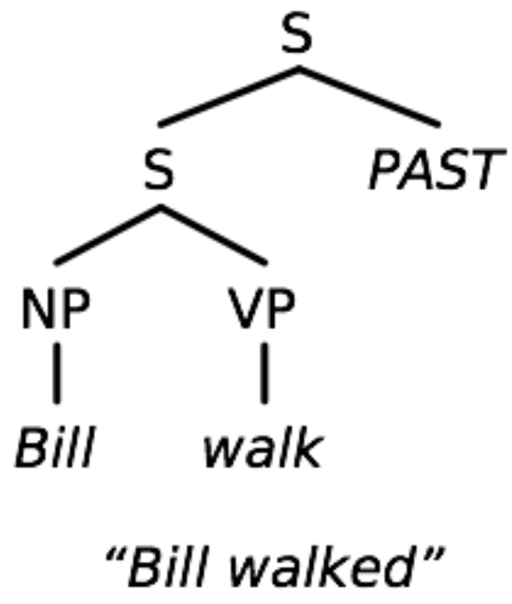
  V an interpretation function

- Definition of overlapping events:

  $e \circ e'$ iff neither $e < e'$ *nor* $e' < e$

Event Semantics:
Tense in Semantic Construction

- Tense is encoded in the verb inflection.
- We can represent inflection as an abstract tense operator commanding the untensed rest of the sentence.
- Semantic representation of tense operators expresses temporal location of the reported event with respect to the utterance event:

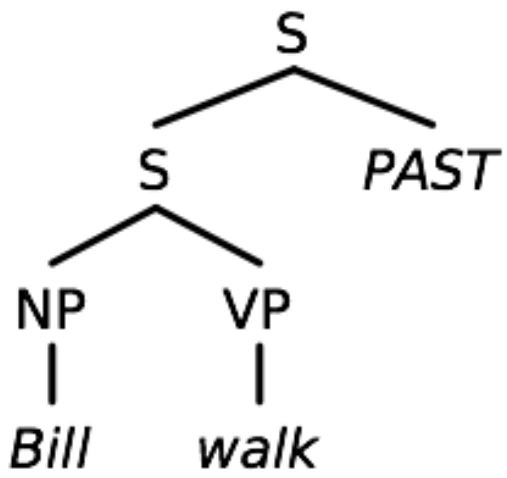


$$\textit{PAST} \Rightarrow \lambda E \exists e(E(e) \wedge e < e_u)\text{:}\ \ \langle\langle e,t\rangle,t\rangle$$

$$\textit{PRES} \Rightarrow \lambda E \exists e(E(e) \wedge e \circ e_u)\text{:}\ \ \langle\langle e,t\rangle,t\rangle$$

Event Semantics:
Tense in Semantic Construction

- Application of the tense operator integrates temporal information and at the same time binds the open event variable:



"Bill walked"

walk $\Rightarrow \lambda x\ \lambda e\ [walk(e, x)]$

Bill walk $\Rightarrow \lambda x\ \lambda e\ [walk(e, x)](b)$

$\Leftrightarrow \lambda e\ [walk(e, b)]$

Bill walk PAST

$\Rightarrow \lambda E\ \exists e[E(e) \wedge e<e_u](\lambda e\ [walk(e, b)])$

$\Leftrightarrow \exists e\ [\lambda e\ [walk(e, b)](e) \wedge e<e_u]$

$\Leftrightarrow \exists e\ [walk(e, b) \wedge e<e_u]$

Time Expressions

- *John arrived at 9 p.m.*
- *The lecture is on Tuesday.*
- *Mozart was born in 1756.*
- *Mary had left two hours, before John arrived.*

Event Structures with Explicit Time Representations

- An temporal event structure:

  $M = \langle U, E, T, <, t_u, tl, V \rangle$,

  U, E, and T non-empty and mutually disjoint,

  < a linear ordering on T

  $t_u \in T$ is the utterance time

  tl a function from E to intervals in T (“temporal location”)

  V an interpretation function

- Definition of event precedence and overlap:

  $e < e'$ iff for all $t \in tl(e)$, $t' \in tl(e')$: $t < t'$

  $e \circ e'$ iff $tl(e) \cap tl(e') \neq \emptyset$