Grammar in TTR: frames and lexical semantics; Incremental context and content; Tense and aspect

Robin Cooper
University of Gothenburg
Jonathan Ginzburg
Université Paris-Diderot, Sorbonne Paris-Cité

Type theory with records for natural language semantics,
NASSLLI 2012
Lecture 2

Outline

Frames in lexical semantics (Cooper, 2010, 2012)

Grammar rules as dependent event types (Cooper, 2012)

Speech events and incremental parsing

Characterizing situation types

Outline

Frames in lexical semantics (Cooper, 2010, 2012)

Grammar rules as dependent event types (Cooper, 2012)

Speech events and incremental parsing

Characterizing situation types

The content of *ran*

- $\lambda r{:}\left[\mathrm{x}{:}\mathit{Ind}\right]\left(\left[\begin{array}{lcl} \text{e-time} & : & \mathit{Time} \\ \mathrm{c}_{\mathrm{tns}} & : & \text{e-time.end} < \iota\text{.start} \\ \mathrm{c}_{\mathrm{run}} & : & \mathrm{run}(r.\mathrm{x},\text{e-time}) \end{array}\right]\right)$

The content of *ran*

- $\lambda r{:}\left[\text{x}{:}\mathit{Ind}\right]\left(\left[\begin{array}{lcl} \text{e-time} & : & \mathit{Time} \\ \text{c}_{\text{tns}} & : & \text{e-time.end} < \iota\text{.start} \\ \text{c}_{\text{run}} & : & \text{run}(r\text{.x,e-time}) \end{array}\right]\right)$
- Frames as arguments

The content of *ran*

- $\lambda r{:}\left[\text{x}{:}\mathit{Ind}\right]\left(\begin{bmatrix} \text{e-time} & : & \mathit{Time} \\ \text{c}_{\text{tns}} & : & \text{e-time.end} < \iota\text{.start} \\ \text{c}_{\text{run}} & : & \text{run}(r\text{.x,e-time}) \end{bmatrix}\right)$
- Frames as arguments
- This is an individual level verb phrase interpretation, even though it takes a frame as argument, the predicate ‘run’ takes an individual as argument.

rises – a predicate of frames

- $\lambda r{:}\begin{bmatrix}\text{x}:\textit{Ind}\end{bmatrix}(\begin{bmatrix}\text{e-time} & : & \textit{TimeInt} \\ \text{c}_{\text{tns}} & : & \text{e-time}=\iota \\ \text{c}_{\text{run}} & : & \text{rise}(r,\text{e-time})\end{bmatrix})$

rises – a predicate of frames

- $\lambda r{:}\begin{bmatrix}\mathrm{x}{:}\mathit{Ind}\end{bmatrix}(\begin{bmatrix}\text{e-time} & : & \mathit{TimeInt} \\ \mathrm{c}_{\mathrm{tns}} & : & \text{e-time}=\iota \\ \mathrm{c}_{\mathrm{run}} & : & \mathrm{rise}(r,\text{e-time})\end{bmatrix})$
- This is a frame level verb phrase interpretation, the predicate ‘rise’ takes a frame as argument

rises – a predicate of frames

- $\lambda r{:}\left[\text{x}{:}\textit{Ind}\right]\left(\begin{bmatrix} \text{e-time} & : & \textit{TimeInt} \\ \text{c}_{\text{tns}} & : & \text{e-time}{=}\ \iota \\ \text{c}_{\text{run}} & : & \text{rise}(r,\text{e-time}) \end{bmatrix}\right)$
- This is a frame level verb phrase interpretation, the predicate 'rise' takes a frame as argument
- *The temperature is rising, the temperature is ninety* $\not\Rightarrow$ *Ninety is rising*

rises – a predicate of frames

- $\lambda r{:}\big[\text{x}{:}\mathit{Ind}\big]\left(\begin{bmatrix} \text{e-time} & : & \mathit{TimeInt} \\ \text{c}_{\text{tns}} & : & \text{e-time}{=}\ \iota \\ \text{c}_{\text{run}} & : & \text{rise}(r,\text{e-time}) \end{bmatrix}\right)$
- This is a frame level verb phrase interpretation, the predicate ‘rise’ takes a frame as argument
- *The temperature is rising, the temperature is ninety* $\not\Rightarrow$ *Ninety is rising*
- What can it mean for a frame to rise?

The record type *AmbTemp*

- $\left[\begin{array}{lll} \mathsf{x} & : & \mathit{Ind} \\ \mathsf{e\text{-}time} & : & \mathit{Time} \\ \mathsf{e\text{-}location} & : & \mathit{Loc} \\ \mathsf{c}_{\mathrm{temp\_at\_in}} & : & \mathsf{temp\_at\_in(e\text{-}time,\ e\text{-}location,\ x)} \end{array}\right]$

The record type *AmbTemp*

- $$\begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{e-time} & : & \mathit{Time} \\ \text{e-location} & : & \mathit{Loc} \\ \text{c}_{\text{temp\_at\_in}} & : & \text{temp\_at\_in(e-time, e-location, x)} \end{bmatrix}$$
- a record of this type will meet the following two conditions:
  - it will contain *at least* fields with the same labels as the type (it may contain more)

The record type *AmbTemp*

- $\left[\begin{array}{lll} \text{x} & : & \mathit{Ind} \\ \text{e-time} & : & \mathit{Time} \\ \text{e-location} & : & \mathit{Loc} \\ \text{c}_{\text{temp\_at\_in}} & : & \text{temp\_at\_in(e-time, e-location, x)} \end{array}\right]$
- a record of this type will meet the following two conditions:
  - it will contain *at least* fields with the same labels as the type (it may contain more)
  - each field in the record with the same label as a field in the record type will contain an object of the type in the corresponding field of the record type. (Any additional fields with different labels to those in the record type may contain objects of any type.)

The type *TempRise*

$$\begin{bmatrix} \text{e-time}: \mathit{TimeInt} \\ \text{start}: \begin{bmatrix} \text{x}: \mathit{Ind} \\ \text{e-time}{=}\text{e-time.start}: \mathit{Time} \\ \text{e-location}: \mathit{Loc} \\ \text{c}_{\text{temp\_at\_in}}: \text{temp\_at\_in(start.e-time, start.e-location, start.x)} \end{bmatrix} \\ \text{end}: \begin{bmatrix} \text{x}: \mathit{Ind} \\ \text{e-time}{=}\text{e-time.end}: \mathit{Time} \\ \text{e-location}{=}\text{start.e-location}: \mathit{Loc} \\ \text{c}_{\text{temp\_at\_in}}: \text{temp\_at\_in(end.e-time, end.e-location, end.x)} \end{bmatrix} \\ \text{event}{=}\text{start}^\frown\text{end}: \mathit{AmbTemp}^\frown\mathit{AmbTemp} \\ \text{c}_{\text{incr}}: \text{start.x}{<}\text{end.x} \end{bmatrix}$$

The type *TempRise*

$$\begin{bmatrix} \text{e-time}:\mathit{TimeInt} \\ \text{start}: \begin{bmatrix} \text{x}:\mathit{Ind} \\ \text{e-time}{=}\text{e-time.start}:\mathit{Time} \\ \text{e-location}:\mathit{Loc} \\ \text{c}_{\text{temp\_at\_in}}:\text{temp\_at\_in(start.e-time, start.e-location, start.x)} \end{bmatrix} \\ \text{end}: \begin{bmatrix} \text{x}:\mathit{Ind} \\ \text{e-time}{=}\text{e-time.end}:\mathit{Time} \\ \text{e-location}{=}\text{start.e-location}:\mathit{Loc} \\ \text{c}_{\text{temp\_at\_in}}:\text{temp\_at\_in(end.e-time, end.e-location, end.x)} \end{bmatrix} \\ \text{event}{=}\text{start}^\frown\text{end}:\mathit{AmbTemp}^\frown\mathit{AmbTemp} \\ \text{c}_{\text{incr}}:\text{start.x}<\text{end.x} \end{bmatrix}$$

If r:*AmbTemp* and i:*TimeInt* then e:rise(r,i) iff e:*TempRise*, e.start=r and e.e-time=i.

Does *rise* have a general meaning?

At this level of detail, the type of an event of a price rise is slightly different.

The type *PriceRise*

$$\left[\begin{array}{l}
\text{e-time}:\textit{TimeInt}\\
\text{start}:\left[\begin{array}{l}
\text{x}:\textit{Ind}\\
\text{e-time=e-time.start}:\textit{Time}\\
\text{e-location}:\textit{Loc}\\
\text{commodity}:\textit{Ind}\\
\text{c}_{\text{price_of_at_in}}:\text{price_of_at_in(start.commodity,}\\
\qquad\qquad\qquad\text{start.e-time, start.e-location, start.x)}
\end{array}\right]\\
\text{end}:\left[\begin{array}{l}
\text{x}:\textit{Ind}\\
\text{e-time=e-time.end}:\textit{Time}\\
\text{e-location=start.e-location}:\textit{Loc}\\
\text{commodity=start.commodity}:\textit{Ind}\\
\text{c}_{\text{price_of_at_in}}:\text{price_of_at_in(end.commodity,}\\
\qquad\qquad\qquad\text{end.e-time, end.e-location, end.x)}
\end{array}\right]\\
\text{event=start}^\frown\text{end}:\textit{Price}^\frown\textit{Price}\\
\text{c}_{\text{incr}}:\text{start.x}<\text{end.x}
\end{array}\right]$$

The Titan rises

(description of a video game)

As they get to deck, they see the Inquisitor, calling out to a Titan in the seas. **The giant Titan rises through the waves**, shrieking at the Inquisitor.

The type *LocRise*

$$\begin{bmatrix}
\text{e-time}:\mathit{TimeInt} \\
\text{start}: \begin{bmatrix} \text{x}:\mathit{Ind} \\ \text{e-time}{=}\text{e-time.start}:\mathit{Time} \\ \text{e-location}:\mathit{Loc} \\ \text{c}_{\text{at}}:\text{at(start.x,start.e-location,start.e-time)} \end{bmatrix} \\
\text{end}: \begin{bmatrix} \text{x}{=}\text{start.x}:\mathit{Ind} \\ \text{e-time}{=}\text{e-time.end}:\mathit{Time} \\ \text{e-location}:\mathit{Loc} \\ \text{c}_{\text{at}}:\text{at(end.x,end.e-location,end.e-time)} \end{bmatrix} \\
\text{event}{=}\text{start}^\frown\text{end}:\mathit{Position}^\frown\mathit{Position} \\
\text{c}_{\text{incr}}:\text{height(start.e-location)}<\text{height(end.e-location)}
\end{bmatrix}$$

Semantic coordination

(Work with Staffan Larsson: Larsson, 2007; Cooper and Larsson, 2009; Larsson and Cooper, 2009)

Instead of "What is the meaning of expression *E*?"

Semantic coordination

(Work with Staffan Larsson: Larsson, 2007; Cooper and Larsson, 2009; Larsson and Cooper, 2009)

Instead of "What is the meaning of expression E?"

we propose

the coordination question
: Given resources R, how can agent A construct a meaning for a particular utterance U of expression E?

Semantic coordination

(Work with Staffan Larsson: Larsson, 2007; Cooper and Larsson, 2009; Larsson and Cooper, 2009)

Instead of "What is the meaning of expression E?"

we propose

the coordination question
: Given resources R, how can agent A construct a meaning for a particular utterance U of expression E?

the resource update question
: What effect will this have on A's resources R?

Outline

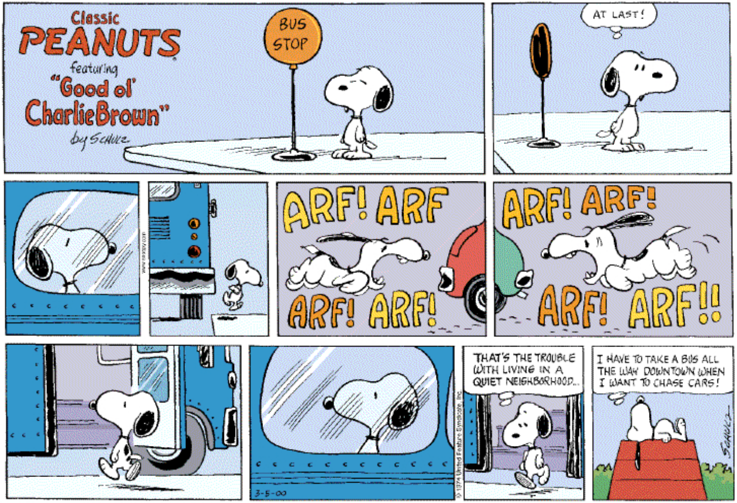
Frames in lexical semantics (Cooper, 2010, 2012)

Grammar rules as dependent event types (Cooper, 2012)

Speech events and incremental parsing

Characterizing situation types

Fernando's string theory



A Fernando finite automaton representation

$\boxed{\text{wait-at-busstop}}^* \; \boxed{\text{bus-arrive}} \; \boxed{\text{get-on-bus}} \; \boxed{\text{travel-on-bus}} \; \boxed{\text{get-off-bus}} \; \boxed{}^+$

String types and finite automata

- $bus\text{-}trip = get\text{-}bus^\frown travel\text{-}on\text{-}bus^\frown get\text{-}off\text{-}bus$

String types and finite automata

- $\textit{bus-trip} = \textit{get-bus}^\frown\textit{travel-on-bus}^\frown\textit{get-off-bus}$
- $\textit{get-bus} = \textit{wait-at-busstop}^{*\frown}\textit{bus-arrive}^\frown\textit{get-on-bus}$

String types and finite automata

- $\textit{bus-trip} = \textit{get-bus}^\frown\textit{travel-on-bus}^\frown\textit{get-off-bus}$
- $\textit{get-bus} = \textit{wait-at-busstop}^{*\frown}\textit{bus-arrive}^\frown\textit{get-on-bus}$

or in Fernando's kind of notation:

- $\mathcal{L}(\textit{bus-trip}) = \mathcal{L}(\textit{get-bus})^\frown\mathcal{L}(\textit{travel-on-bus})^\frown\mathcal{L}(\textit{get-off-bus})$
- $\mathcal{L}(\textit{get-bus}) = \mathcal{L}(\textit{wait-at-busstop})^{*\frown}\mathcal{L}(\textit{bus-arrive})^\frown\mathcal{L}(\textit{get-on-bus})$

where $L_1{}^\frown L_2 = \{a^\frown b \mid a \in L_1 \text{ and } b \in L_2\}$

Two kinds of partiality

- not all frames are perceived – infer more

Two kinds of partiality

- not all frames are perceived – infer more
- individual frame type partially specified

Two kinds of partiality

- not all frames are perceived – infer more
- individual frame type partially specified
- $$\begin{bmatrix} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{y} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \text{bus-stop(y)} \\ \mathrm{c}_2 & : & \text{near(x,y)} \\ \mathrm{c}_3 & : & \text{wait-for(x,} \begin{bmatrix} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \text{bus(x)} \\ \mathrm{c}_2 & : & \text{arrive-at(x,y)} \end{bmatrix}\text{)} \end{bmatrix}$$

Specification with Snoopy

$$\left[\begin{array}{lll} \mathrm{x}{=}\mathrm{snoopy} & : & \mathit{Ind} \\ \mathrm{y} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \text{bus-stop(y)} \\ \mathrm{c}_2 & : & \text{near(x,y)} \\ \mathrm{c}_3 & : & \text{wait-for(x,} \left[\begin{array}{lll} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c}_1 & : & \text{bus(x)} \\ \mathrm{c}_2 & : & \text{arrive-at(x,y)} \end{array}\right]\text{)} \end{array}\right]$$

What are *manifest fields* $\left[\ \ell{=}a \quad : \quad T\ \right]$?

Manifest fields

$\left[\ \ell{=}a \quad : \quad T\ \right]$

is a convenient notation for

$\left[\ \ell \quad : \quad T_a\ \right]$

If $a : T$, then T_a is a type (the *singleton type of* a).

$b : T_a$ iff $b = a$

Partial perception of linguistic events

- perceive phonology (or phonetics)
- infer syntax/semantics

Grammar rules

Grammar rules are of the form

$\lambda s_1 : T_1 \ldots \lambda s_n : T_n(T)$

where T_i and T are *sign types*.

Sign types correspond to the notion of sign in HPSG.

The type *Sign*

$$\begin{bmatrix} \text{s-event} & : & \mathit{SEvent} \\ \text{synsem} & : & \begin{bmatrix} \text{cat} & : & \mathit{Cat} \\ \text{cnt} & : & \mathit{Cnt} \end{bmatrix} \end{bmatrix}$$

But what are *SEvent*, *Cat* and *Cnt*?

The type *SEvent*

$$\begin{bmatrix} \text{phon} & : & \textit{Phon} \\ \text{s-time} & : & \begin{bmatrix} \text{start} & : & \textit{Time} \\ \text{end} & : & \textit{Time} \end{bmatrix} \\ \text{utt}_{\text{at}} & : & \text{uttered\_at(phon, s-time.start, s-time.end)} \end{bmatrix}$$

A minimal solution, from Cooper (2012).

The type *SEvent*, a more refined type

$$\begin{bmatrix} \text{e-time} & : & \mathit{TimeInt} \\ \text{e-loc} & : & \mathit{Loc} \\ \text{sp} & : & \mathit{Ind} \\ \text{au} & : & \mathit{Ind} \\ \text{phon} & : & \mathit{Phon} \\ \text{e} & : & \text{utter(sp,phon,au,e-time,e-loc)} \end{bmatrix}$$

However, this may not always be correct: more than one person may be in the audience; more than one person may collaborate on a single speech event - *split utterances*.

The type *Cat*

s, np, vp, $\mathrm{n_{prop}}$, $\mathrm{v_i}$: *Cat*

There will be more in a more extensive fragment.

The type *Cnt*

$RecType \vee (Ppty \vee Quant)$

There will be more in a more extensive fragment

The type *Cnt*

$RecType \vee (Ppty \vee Quant)$

There will be more in a more extensive fragment

If T_1, T_2 are types, then $T_1 \vee T_2$ is a type (the *join* of T_1 and T_2).
$a : T_1 \vee T_2$ iff either $a : T_1$ or $a : T_2$

The type *Cnt*

$RecType \vee (Ppty \vee Quant)$

There will be more in a more extensive fragment

If T_1, T_2 are types, then $T_1 \vee T_2$ is a type (the *join* of T_1 and T_2).
$a : T_1 \vee T_2$ iff either $a : T_1$ or $a : T_2$

Ppty, "property" is to be $\left[\text{x}:Ind\right] \rightarrow RecType$
(cf. $\langle e, t \rangle$)

The type *Cnt*

$RecType \vee (Ppty \vee Quant)$

There will be more in a more extensive fragment

If T_1, T_2 are types, then $T_1 \vee T_2$ is a type (the *join* of T_1 and T_2).
$a : T_1 \vee T_2$ iff either $a : T_1$ or $a : T_2$

Ppty, "property" is to be $\left[\text{x}{:}Ind\right] \rightarrow RecType$
(cf. $\langle e, t \rangle$)

Quant, "quantifier" is to be $Ppty \rightarrow RecType$
(cf. $\langle\langle e, t \rangle, t \rangle$)

The lexicon: Proper names

If W is a phonological type such as “Sam” or “John” and $a{:}\mathit{Ind}$, $\mathrm{lex}_{\mathrm{n_{Prop}}}(W, a)$ is

$$\mathit{Sign} \wedge\!\!\!\!. \left[\begin{array}{lll} \text{s-event} & : & \left[\begin{array}{lll} \text{phon} & : & W \end{array}\right] \\ \text{synsem} & : & \left[\begin{array}{lll} \text{cat}{=}\mathrm{n_{Prop}} & : & \mathit{Cat} \\ \text{cnt}{=}\lambda v{:}\mathit{Ppty}(v(\left[\text{x}{=}a\right])) & : & \mathit{Quant} \end{array}\right] \end{array}\right]$$

$T_1 \wedge\!\!\!\!. \, T_2$ is the *merge* of the two types T_1, T_2. If at least on of the two types is not a record type it is identical with the meet $T_1 \wedge T_2$. So what is the meet of two types and the merge of two record types?

Meets and merges

If T_1, T_2 are types, then $T_1 \wedge T_2$ is a type (the *meet* of T_1 and T_2).
$a : T_1 \wedge T_2$ iff $a : T_1$ and $a : T_2$

The basic idea of *merge* (full definition in Cooper, 2012):

$$\left[\mathrm{f}{:}\,T_1\right] \dot{\wedge} \left[\mathrm{g}{:}\,T_2\right] = \begin{bmatrix} \mathrm{f}{:}\,T_1 \\ \mathrm{g}{:}\,T_2 \end{bmatrix}$$

$$\left[\mathrm{f}{:}\,T_1\right] \dot{\wedge} \left[\mathrm{f}{:}\,T_2\right] = \left[\mathrm{f}{:}\,T_1 \dot{\wedge} T_2\right]$$

cf. unification

The lexicon: intransitive verbs (individual level)

If W is a phonological type like "run" or "walk" and p is a predicate with arity $\langle \mathit{Ind}, \mathit{TimeInt}\rangle$, $\mathrm{lex}_{\mathrm{V_i}}(W, p)$ is

$$\mathit{Sign} \mathbin{\dot\wedge} \begin{bmatrix} \text{s-event} : \begin{bmatrix} \text{phon} : W \end{bmatrix} \\ \text{synsem} : \begin{bmatrix} \text{cat}{=}\mathrm{v_i} : \mathit{Cat} \\ \text{cnt}{=}\lambda r{:}\begin{bmatrix} \text{x} : \mathit{Ind} \end{bmatrix}\ (\begin{bmatrix} \text{e-time} : \mathit{TimeInt} \\ \text{c}_W : p(r.\text{x}, \text{e-time}) \end{bmatrix}) : \mathit{Ppty} \end{bmatrix} \end{bmatrix}$$

Composition rules as dependent types

unary rules $\lambda s : T_1(T_2)$, where $T_1, T_2 \sqsubseteq Sign$

binary rules $\lambda s : T_1 \frown T_2(T_3)$, where $T_1, T_2, T_3 \sqsubseteq Sign$

We need to address subtyping and concatenation.

Subtyping

$T_1 \sqsubseteq T_2$ just in case $\{a \mid a : T_1\} \subseteq \{a \mid a : T_2\}$ *for all assignments to the basic types*.

Some examples:

- $\begin{bmatrix} \text{f}: T_1 \\ \text{g}: T_2 \end{bmatrix} \sqsubseteq \begin{bmatrix} \text{f}: T_1 \end{bmatrix}$
- $\begin{bmatrix} \text{f}{=}a: T \end{bmatrix} \sqsubseteq \begin{bmatrix} \text{f}: T \end{bmatrix}$
- $T_1 \wedge T_2 \sqsubseteq T_1$
- $T_1 \sqsubseteq T_1 \vee T_2$

Temporal concatenation of signs

1. for any T_1, $T_2 \sqsubseteq \mathit{Sign}$, $T_1{}^\frown{}^{\mathrm{temp}}T_2 \in \mathbf{Type}$
2. $s : T_1{}^\frown{}^{\mathrm{temp}}T_2$ iff $s = s_1{}^\frown s_2$, $s_1 : T_1$, $s_2 : T_2$ and s_1.s-event.s-time.end $<$ s_2.s-event.s-time.start.

This differs from the temporal concatenation in Lecture 1 in that we make explicit reference to s-event and s-time within signs.

Rule components I

unary_sign	$\lambda s{:}Sign(Sign)$
binary_sign	$\lambda s{:}Sign^{\frown\text{temp}}Sign(Sign)$
phon_id	$\lambda s{:}\left[\text{s-event}{:}\left[\text{phon}{:}Phon\right]\right]$ $\left(\left[\text{s-event}{:}\left[\text{phon}{=}s.\text{s-event.phon}{:}Phon\right]\right]\right)$
phon_concat	$\lambda s{:}\left[\text{s-event}{:}\left[\text{phon}{:}Phon\right]\right]^{\frown}\left[\text{s-event}{:}\left[\text{phon}{:}Phon\right]\right]$ $\left(\left[\text{s-event}{:}\left[\text{phon}{=}s[1].\text{s-event.phon}^{\frown}s[2].\text{s-event.phon}{:}Phon\right]\right]\right)$
unary_cat	$\lambda c_1{:}Cat(\lambda c_2{:}Cat(\lambda s{:}\left[\text{cat}{=}c_1{:}Cat\right](\left[\text{cat}{=}c_2{:}Cat\right])))$
binary_cat	$\lambda c_1{:}Cat(\lambda c_2{:}Cat(\lambda c_3{:}Cat$ $(\lambda s{:}\left[\text{cat}{=}c_1{:}Cat\right]^{\frown}\left[\text{cat}{=}c_2{:}Cat\right](\left[\text{cat}{=}c_3{:}Cat\right])))$
cnt_id	$\lambda s{:}\left[\text{synsem}{:}\left[\text{cnt}{:}Cnt\right]\right]$ $\left(\left[\text{synsem}{:}\left[\text{cnt}{=}s.\text{synsem.cnt}{:}Cnt\right]\right]\right)$

Rule components II

$$\text{cnt\_forw\_app} \quad \lambda T_1{:}\mathit{Type}(\lambda T_2{:}\mathit{Type}\;(\lambda s{:}\left[\text{synsem}{:}\left[\text{cnt}{:}T_1 \rightarrow T_2\right]\right]^\frown\left[\text{synsem}{:}\left[\text{cnt}{:}T_1\right]\right]\;(\left[\text{synsem}{:}\left[\text{cnt}{=}s[1].\text{synsem.cnt}(s[2].\text{synsem.cnt}){:}T_2\right]\right])))$$

$$\text{fin\_id} \quad \lambda s{:}\left[\text{fin}{:}\mathit{Bool}\right](\left[\text{fin}{=}s.\text{fin}{:}\mathit{Bool}\right])$$

$$\text{fin\_hd} \quad \lambda s{:}\mathit{Sign}^\frown\left[\text{fin}{=}1{:}\mathit{Bool}\right](\left[\text{fin}{=}s.\text{fin}{:}\mathit{Bool}\right])$$

Merging functions

1. $\lambda v{:}T_1(T_2)\ \dot{\wedge}\ \lambda v{:}T_3(T_4)$ is to be $\lambda v{:}T_1 \dot{\wedge} T_3(T_2 \dot{\wedge} T_4)$
2. $\lambda v{:}T_1 ^\frown T_2(T_3)\ \dot{\wedge}\ \lambda v{:}T_4 ^\frown T_5(T_6)$ is to be $\lambda v{:}(T_1 \dot{\wedge} T_4)^\frown(T_2 \dot{\wedge} T_5)\ (T_3 \dot{\wedge} T_6)$
3. $\lambda v{:}T_1 ^{\frown temp} T_2(T_3)\ \dot{\wedge}\ \lambda v{:}T_4 ^\frown T_5(T_6)$ is to be $\lambda v{:}(T_1 \dot{\wedge} T_4)^{\frown temp}(T_2 \dot{\wedge} T_5)\ (T_3 \dot{\wedge} T_6)$

Composition rules

S $\rightarrow$ NP VP binary_sign $\dot{\wedge}$ phon_concat $\dot{\wedge}$ binary_cat(np)(vp)(s) $\dot{\wedge}$ fin_hd $\dot{\wedge}$ cnt_forw_app(*Ppty*)(*RecType*)

NP $\rightarrow$ N unary_sign $\dot{\wedge}$ phon_id $\dot{\wedge}$ unary_cat($\mathrm{n}_{\mathrm{Prop}}$)(np) $\dot{\wedge}$ cnt_id

VP $\rightarrow$ V_i unary_sign $\dot{\wedge}$ phon_id $\dot{\wedge}$ unary_cat(v_i)(vp) $\dot{\wedge}$ fin_id $\dot{\wedge}$ cnt_id

Why bother?

- We want a detailed account of how syntactic rules can be related to perception of events and reasoning about them
- We want ultimately to explain how agents construct new composition rules

Outline

Frames in lexical semantics (Cooper, 2010, 2012)

Grammar rules as dependent event types (Cooper, 2012)

Speech events and incremental parsing

Characterizing situation types

Parsing

e_1

Parsing

$$\begin{array}{c} T_1 \\ | \\ e_1 \end{array}$$

Parsing

$$\begin{array}{ccc} T_1 & \qquad\qquad & \\ | & & | \\ e_1 & & e_2 \end{array}$$

Parsing

$$\begin{array}{ccc} T_1 & \qquad & T_2 \\ | & & | \\ e_1 & & e_2 \end{array}$$

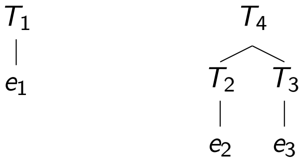
Parsing

$$\begin{array}{ccc} T_1 & T_2 & \\ | & | & | \\ e_1 & e_2 & e_3 \end{array}$$

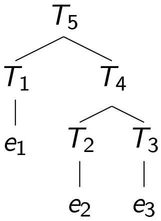
Parsing

$$\begin{array}{ccccc} T_1 & \quad & T_2 & \quad & T_3 \\ | & & | & & | \\ e_1 & & e_2 & & e_3 \end{array}$$

Parsing



Parsing



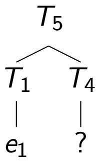
Parsing with left-corner predictions

$$\begin{array}{c} | \\ e_1 \end{array}$$

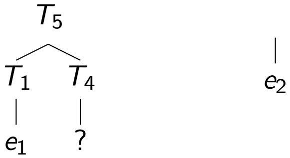
Parsing with left-corner predictions

$$
\begin{array}{c}
T_1 \\
| \\
e_1
\end{array}
$$

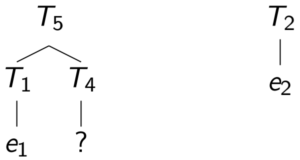
Parsing with left-corner predictions



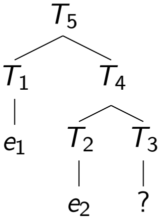
Parsing with left-corner predictions



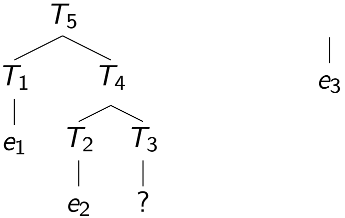
Parsing with left-corner predictions



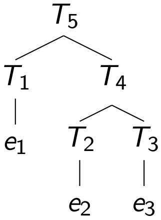
Parsing with left-corner predictions



Parsing with left-corner predictions



Parsing with left-corner predictions



Parsing with sign types

|
Snoopy

Parsing with sign types

$$\begin{array}{c}\left[\begin{array}{ll}\text{s-event}: & \left[\begin{array}{l}\text{phon}{=}\text{“Snoopy”}:\mathit{Phon}\\ \text{s-time}:\mathit{TimeInt}\\ \text{ref}{=}\text{snoopy}:\mathit{Ind}\\ \text{c}_{\mathit{utt}}:\text{uttered(phon,s-time)}\\ \text{c}_{\mathit{ref}}:\text{ref(phon,ref)}\end{array}\right]\\ \text{synsem}: & \left[\begin{array}{l}\text{cat}:\mathit{Cat}\\ \text{cnt}:\mathit{Cnt}\end{array}\right]\end{array}\right]\\ |\\ \mathit{Snoopy}\end{array}$$

Parsing with sign types

$$\left[\begin{array}{ll} \text{s-event:} & \left[\begin{array}{l} \text{phon}{=}\text{“Snoopy”}:\mathit{Phon} \\ \text{s-time}:\mathit{TimeInt} \\ \text{ref}{=}\text{snoopy}:\mathit{Ind} \\ \text{c}_{\mathit{utt}}:\text{uttered(phon,s-time)} \\ \text{c}_{\mathit{ref}}:\text{ref(phon,ref)} \end{array}\right] \\ \text{synsem:} & \left[\begin{array}{l} \text{cat}{=}\text{np}:\mathit{Cat} \\ \text{cnt}:\mathit{Cnt} \end{array}\right] \end{array}\right]$$

$$|$$

Snoopy

Parsing with sign types

$$\left[\begin{array}{l}\text{s-event}:\left[\begin{array}{l}\text{phon}=\text{“Snoopy”}:\mathit{Phon}\\ \text{s-time}:\mathit{TimeInt}\\ \text{ref}=\text{snoopy}:\mathit{Ind}\\ \text{c}_{\mathit{utt}}:\text{uttered(phon,s-time)}\\ \text{c}_{\mathit{ref}}:\text{ref(phon,ref)}\end{array}\right]\\ \text{synsem}:\left[\begin{array}{l}\text{cat}=\text{np}:\mathit{Cat}\\ \text{cnt}=\text{npsem(“Snoopy”,snoopy)}:\mathit{Cnt}\end{array}\right]\end{array}\right]$$

$$\begin{array}{c}|\\ \mathit{Snoopy}\end{array}$$

Parsing with sign types

$$\left[\begin{array}{l}\text{s-event}:\mathit{SEvent}\\ \text{synsem}:\left[\begin{array}{l}\text{cat}{=}\text{s}:\mathit{Cat}\\ \text{cnt}:\mathit{Cnt}\end{array}\right]\end{array}\right]$$

$$\left[\begin{array}{l}\text{s-event}:\left[\begin{array}{l}\text{phon}{=}\text{“Snoopy”}:\mathit{Phon}\\ \text{s-time}:\mathit{TimeInt}\\ \text{ref}{=}\text{snoopy}:\mathit{Ind}\\ \text{c}_{\mathit{utt}}:\text{uttered(phon,s-time)}\\ \text{c}_{\mathit{ref}}:\text{ref(phon,ref)}\end{array}\right]\\ \text{synsem}:\left[\begin{array}{l}\text{cat}{=}\text{np}:\mathit{Cat}\\ \text{cnt}{=}\text{npsem(“Snoopy”,snoopy)}:\mathit{Cnt}\end{array}\right]\end{array}\right] \qquad \left[\begin{array}{l}\text{s-event}:\mathit{SEvent}\\ \text{synsem}:\left[\begin{array}{l}\text{cat}{=}\text{vp}:\mathit{Cat}\\ \text{cnt}:\mathit{Cnt}\end{array}\right]\end{array}\right]$$

Snoopy

Perception, inference and structure

- The kind of processing we have talked about require structured objects which can be modified

Perception, inference and structure

- The kind of processing we have talked about require structured objects which can be modified
- We have talked about types of strings, records and trees

Perception, inference and structure

- The kind of processing we have talked about require structured objects which can be modified
- We have talked about types of strings, records and trees
- The types themselves are articulated into components so that we can derive new types from old.

Structure and proof theory

- Classical model theoretic treatments are not structured enough or at least not in the right way

Structure and proof theory

- Classical model theoretic treatments are not structured enough or at least not in the right way
- We have chosen to pursue a semantic approach with much more finely structured objects

Structure and proof theory

- Classical model theoretic treatments are not structured enough or at least not in the right way
- We have chosen to pursue a semantic approach with much more finely structured objects
- An alternative is to assume that there is an abstract language associated with a proof theory which exploits its syntactic structure

Outline

The (Aristotle-Ryle-Kenny-)Vendler classification I

Some classic linguistic references: Dowty (1979), Smith (1991)
More a classification of sentences than verbs (Verkuyl, 1989).
Bach (1986a,b) introduced the term *eventuality*. A more recent overview of this extensive literature: Steedman (2005).

states true over a time interval, no non-homogeneous subevents, subinterval property (for all subintervals?): *know the answer, believe a proposition, have a dog, desire a better job, love a friend*

activities true over a time interval, non-homogeneous subevents, subinterval property (for some subintervals?): *run, walk, swim, push a cart, drive a car*

The (Aristotle-Ryle-Kenny-)Vendler classification II

accomplishments true over a time interval, non-homogeneous subevents, no subinterval property, culmination (Moens and Steedman, 1988): *paint a picture, make a chair, deliver a sermon, draw a circle, push a cart to the barn, recover from illness*

achievements true at a time point (punctual), no subevents, no subinterval property, just a culmination (?): *recognize a friend, spot a filmstar, find a coin, lose one's pen, reach the summit, die*

Eventualities

- T is an *eventuality* type iff $T \sqsubseteq \begin{bmatrix} \text{e-time}: \mathit{TimeInt} \\ \text{ev}: \sigma(\text{e-time}) \end{bmatrix}$ for some σ such that for any t: *TimeInt*, $\sigma(t)$ is a ptype.

Eventualities

- T is an *eventuality* type iff $T \sqsubseteq \begin{bmatrix} \text{e-time}: \mathit{TimeInt} \\ \text{ev}: \sigma(\text{e-time}) \end{bmatrix}$ for some σ such that for any t: *TimeInt*, $\sigma(t)$ is a ptype.
- Examples:
  - $\begin{bmatrix} \text{e-time}: \mathit{TimeInt} \\ \text{ev}: \text{hug}(b, d, \text{e-time}) \end{bmatrix}$

Eventualities

- T is an *eventuality* type iff $T \sqsubseteq \begin{bmatrix} \text{e-time}: \mathit{TimeInt} \\ \text{ev}: \sigma(\text{e-time}) \end{bmatrix}$ for some σ such that for any t: *TimeInt*, $\sigma(t)$ is a ptype.
- Examples:
  - $\begin{bmatrix} \text{e-time}: \mathit{TimeInt} \\ \text{ev}: \text{hug}(b, d, \text{e-time}) \end{bmatrix}$
  - $\begin{bmatrix} \text{e-time}: \mathit{TimeInt} \\ \text{x}: \mathit{Ind} \\ \text{c}_{\text{boy}}: \text{boy}(\text{x}, \text{e-time}) \\ \text{y}: \mathit{Ind} \\ \text{c}_{\text{dog}}: \text{dog}(\text{y}, \text{e-time}) \\ \text{ev}: \text{hug}(\text{x}, \text{y}, \text{e-time}) \end{bmatrix}$

States

- An eventuality type $T \sqsubseteq \begin{bmatrix} \text{e-time}: \mathit{TimeInt} \\ \text{ev}: \sigma(\text{e-time}) \end{bmatrix}$ is a *state* type iff

  for any t: *TimeInt*
  $e : \sigma(t)$ iff
  1. $e : T^{+\leq=}$
  2. $\text{first}(e)\text{:}\begin{bmatrix}\text{e-time}\text{:}\begin{bmatrix}\text{start}{=}t.\text{start}\text{:}\mathit{Time}\end{bmatrix}\end{bmatrix}$
  3. $\text{last}(e)\text{:}\begin{bmatrix}\text{e-time}\text{:}\begin{bmatrix}\text{end}{=}t.\text{end}\text{:}\mathit{Time}\end{bmatrix}\end{bmatrix}$

States

- An eventuality type $T \sqsubseteq \begin{bmatrix}\text{e-time}:\mathit{TimeInt}\\ \text{ev}:\sigma(\text{e-time})\end{bmatrix}$ is a *state* type iff
  for any t:*TimeInt*
  $e : \sigma(t)$ iff
  1. $e : T^{+\leq=}$
  2. $\text{first}(e)\text{:}\big[\text{e-time:}\big[\text{start}{=}t\text{.start:}\mathit{Time}\big]\big]$
  3. $\text{last}(e)\text{:}\big[\text{e-time:}\big[\text{end}{=}t\text{.end:}\mathit{Time}\big]\big]$
- Example: e:love(b,d,t) iff e:$\begin{bmatrix}\text{e-time}:\mathit{TimeInt}\\ \text{ev}:\text{love}(b,d,\text{e-time})\end{bmatrix}^{+\leq=}$,
  starting at the beginning of t and ending at the end of t

The subinterval property on states

$$\frac{\text{love}(b,d,t_1)}{\begin{array}{ccc} \text{love}(b,d,t_{1.1}) \ldots & & \text{love}(b,d,t_{1.n}) \\ \text{love}(b,d,t_{1.1.1}) \ldots \quad \text{love}(b,d,t_{1.1.m}) & \ldots & \text{love}(b,d,t_{1.n.1}) \ldots \quad \text{love}(b, \end{array}}$$

no gaps

Activities

- An eventuality type $T \sqsubseteq \begin{bmatrix} \text{e-time}: \textit{TimeInt} \\ \text{ev}: \sigma(\text{e-time}) \end{bmatrix}$ is an *activity* type iff for any t: *TimeInt*
  $e : \sigma(t)$ iff
  1. $e : T^{+\leq}$ or $e : T'$ for some string type T' pointwise distinct from T
  2. $\text{first}(e)$: $\left[\text{e-time}: \left[\text{start}{=}t.\text{start}: \textit{Time}\right]\right]$
  3. $\text{last}(e)$: $\left[\text{e-time}: \left[\text{end}{=}t.\text{end}: \textit{Time}\right]\right]$

Pointwise distinctness

A string type $T_1{}^{\frown\leq} \ldots {}^{\frown\leq} T_n$ is *pointwise distinct from* T iff for all $1 \leq i \leq n$ $T_i \not\sqsubseteq T$.

Subinterval property on activities

Subinterval property on activities

- There can be gaps ($^\frown\leq$)

Subinterval property on activities

- There can be gaps ($\frown_{\leq}$)
- Can bottom out (the second clause with pointwise distinctness)

Subinterval property on activities

- There can be gaps ($\frown^{\leq}$)
- Can bottom out (the second clause with pointwise distinctness)
- Example: Kim runs from time t_1 to t_2.

Subinterval property on activities

- There can be gaps ($\frown^{\leq}$)
- Can bottom out (the second clause with pointwise distinctness)
- Example: Kim runs from time t_1 to t_2.
  - can be periods of running between t_1 and t_2

Subinterval property on activities

- There can be gaps ($\frown^{\leq}$)
- Can bottom out (the second clause with pointwise distinctness)
- Example: Kim runs from time t_1 to t_2.
  - can be periods of running between t_1 and t_2
  - can be periods of non-running between t_1 and t_2

Subinterval property on activities

- There can be gaps ($\frown^{\leq}$)
- Can bottom out (the second clause with pointwise distinctness)
- Example: Kim runs from time t_1 to t_2.
  - can be periods of running between t_1 and t_2
  - can be periods of non-running between t_1 and t_2
  - can bottom out into picking up left foot, moving leg forward etc. i.e. a string of events which does not include a running event (pointwise distinct)

Accomplishments

- An eventuality type $T \sqsubseteq \begin{bmatrix} \text{e-time}: \mathit{TimeInt} \\ \text{ev}: \sigma(\text{e-time}) \end{bmatrix}$ is an *accomplishment* type iff for any $t{:}\mathit{TimeInt}$
  $e : \sigma(t)$ iff
  1. $e : T'$ for some string type T' pointwise distinct from T
  2. $\text{first}(e){:}\begin{bmatrix}\text{e-time}{:}\begin{bmatrix}\text{start}{=}t.\text{start}{:}\mathit{Time}\end{bmatrix}\end{bmatrix}$
  3. $\text{last}(e){:}\begin{bmatrix}\text{e-time}{:}\begin{bmatrix}\text{end}{=}t.\text{end}{:}\mathit{Time}\end{bmatrix}\end{bmatrix}$

Accomplishments

- An eventuality type $T \sqsubseteq \begin{bmatrix}\text{e-time}:\mathit{TimeInt}\\ \text{ev}:\sigma(\text{e-time})\end{bmatrix}$ is an *accomplishment* type iff for any $t{:}\mathit{TimeInt}$ $e : \sigma(t)$ iff
  1. $e : T'$ for some string type T' pointwise distinct from T
  2. $\text{first}(e){:}\big[\text{e-time}{:}\big[\text{start}{=}t.\text{start}{:}\mathit{Time}\big]\big]$
  3. $\text{last}(e){:}\big[\text{e-time}{:}\big[\text{end}{=}t.\text{end}{:}\mathit{Time}\big]\big]$
- no subinterval property – bottoms out immediately

Accomplishments

- An eventuality type $T \sqsubseteq \begin{bmatrix} \text{e-time}: \mathit{TimeInt} \\ \text{ev}: \sigma(\text{e-time}) \end{bmatrix}$ is an *accomplishment* type iff for any t: *TimeInt*
  $e : \sigma(t)$ iff
  1. $e : T'$ for some string type T' pointwise distinct from T
  2. $\text{first}(e) : [\text{e-time}: [\text{start}{=}t.\text{start}: \mathit{Time}]]$
  3. $\text{last}(e) : [\text{e-time}: [\text{end}{=}t.\text{end}: \mathit{Time}]]$
- no subinterval property – bottoms out immediately
- Example: build a house – there is not subevent which is a building of the same house. Any complete building of a house which takes place during the time of this event would have to be a distinct event.

Achievements

- An eventuality type $T \sqsubseteq \begin{bmatrix}\text{e-time}:\mathit{TimeInt}\\ \text{ev}:\sigma(\text{e-time})\end{bmatrix}$ is an *achievement* type iff for any t:*TimeInt*
  $e : \sigma(t)$ iff
  1. $e : T_1{}^{\frown\leq=}T_2$ for some non-proper string types T_1 and T_2 distinct from T
  2. $\text{first}(e):\big[\text{e-time}:\big[\text{start}{=}t.\text{start}:\mathit{Time}\big]\big]$
  3. $\text{last}(e):\big[\text{e-time}:\big[\text{end}{=}t.\text{end}:\mathit{Time}\big]\big]$

Achievements

- An eventuality type $T \sqsubseteq \begin{bmatrix} \text{e-time}: \mathit{TimeInt} \\ \text{ev}: \sigma(\text{e-time}) \end{bmatrix}$ is an *achievement* type iff for any $t{:}\mathit{TimeInt}$
  $e : \sigma(t)$ iff
  1. $e : T_1{}^{\frown\leq=} T_2$ for some non-proper string types T_1 and T_2 distinct from T
  2. $\text{first}(e){:}\left[\text{e-time}{:}\left[\text{start}{=}t.\text{start}{:}\mathit{Time}\right]\right]$
  3. $\text{last}(e){:}\left[\text{e-time}{:}\left[\text{end}{=}t.\text{end}{:}\mathit{Time}\right]\right]$
- we do not require punctuality, just one change without a gap

Achievements

- An eventuality type $T \sqsubseteq \begin{bmatrix} \text{e-time}: \mathit{TimeInt} \\ \text{ev}: \sigma(\text{e-time}) \end{bmatrix}$ is an *achievement* type iff for any t: *TimeInt*
  $e : \sigma(t)$ iff
  1. $e : T_1 {}^\frown{}^{\leq=} T_2$ for some non-proper string types T_1 and T_2 distinct from T
  2. $\text{first}(e) : \begin{bmatrix} \text{e-time} : \begin{bmatrix} \text{start}{=}t.\text{start} : \mathit{Time} \end{bmatrix} \end{bmatrix}$
  3. $\text{last}(e) : \begin{bmatrix} \text{e-time} : \begin{bmatrix} \text{end}{=}t.\text{end} : \mathit{Time} \end{bmatrix} \end{bmatrix}$
- we do not require punctuality, just one change without a gap
- Example: reaching the summit involves not being at the summit, then immediately afterwards being at the top

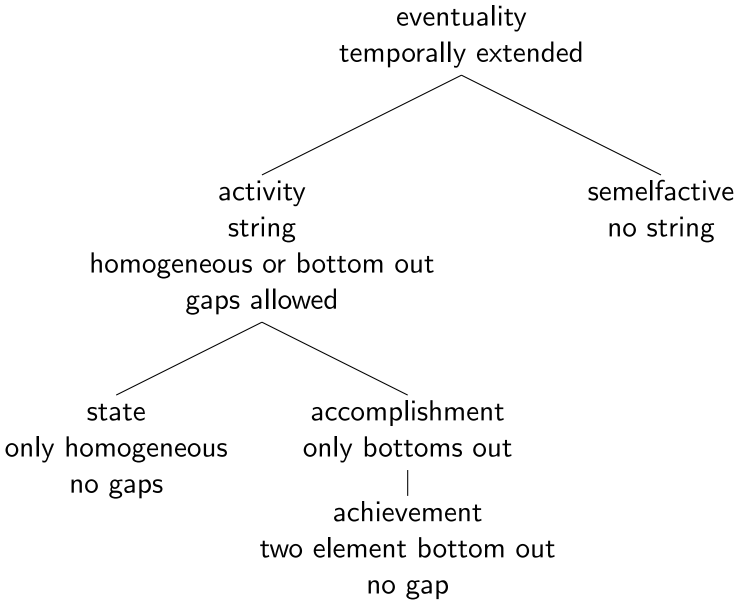
Semelfactives

- An eventuality type $T \sqsubseteq \begin{bmatrix} \text{e-time}: \mathit{TimeInt} \\ \text{ev}: \sigma(\text{e-time}) \end{bmatrix}$ is an *semelfactive* type iff for any $t{:}\mathit{TimeInt}$
  $e : \sigma(t)$ implies e is not a proper string (i.e. not a string with more than one element)

Semelfactives

- An eventuality type $T \sqsubseteq \begin{bmatrix} \text{e-time}: \mathit{TimeInt} \\ \text{ev}: \sigma(\text{e-time}) \end{bmatrix}$ is an *semelfactive* type iff for any t: *TimeInt*
  $e : \sigma(t)$ implies e is not a proper string (i.e. not a string with more than one element)
- Example: a single cough

An unusual eventuality hierarchy



Summary

- Frames in lexical semantics: coordination and flux
- Grammar rules in terms of events: dependent event types
- Parsing and event perception
- Situation types used in aspectual analysis (aktionsart)

Bibliography I

Bach, Emmon (1986a) Natural language metaphysics, in R. Barcan Marcus, G. J. W. Dorn and P. Weingarter (eds.), *Logic, Methodology and Philosophy of Science, VII*, pp. 573–595, Elsevier.

Bach, Emmon (1986b) The Algebra of Events, *Linguistics and Philosophy*, Vol. 9, pp. 5–16.

Cooper, Robin (2010) Frames in formal semantics, in H. Loftsson, E. Rögnvaldsson and S. Helgadóttir (eds.), *IceTAL 2010*, Springer Verlag.

Bibliography II

Cooper, Robin (2012) Type Theory and Semantics in Flux, in R. Kempson, N. Asher and T. Fernando (eds.), *Handbook of the Philosophy of Science*, Vol. 14: Philosophy of Linguistics, Elsevier BV. General editors: Dov M. Gabbay, Paul Thagard and John Woods.

Cooper, Robin and Staffan Larsson (2009) Compositional and ontological semantics in learning from corrective feedback and explicit definition, in J. Edlund, J. Gustafson, A. Hjalmarsson and G. Skantze (eds.), *Proceedings of DiaHolmia: 2009 Workshop on the Semantics and Pragmatics of Dialogue*, pp. 59–66.

Dowty, David R. (1979) *Word Meaning and Montague Grammar: the semantics of verbs and times in generative semantics and in Montague's PTQ*, Reidel Publishing Company.

Bibliography III

Larsson, Staffan (2007) A general framework for semantic plasticity and negotiation, in H. Bunt and E. C. G. Thijsse (eds.), *Proceedings of the 7th International Workshop on Computational Semantics (IWCS-7)*, pp. 101–117.

Larsson, Staffan and Robin Cooper (2009) Towards a formal view of corrective feedback, in A. Alishahi, T. Poibeau and A. Villavicencio (eds.), *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, pp. 1–9.

Moens, Marc and Mark Steedman (1988) Temporal Ontology and Temporal Reference, *Computational Linguistics*, Vol. 14, No. 2, pp. 15–28.

Smith, Carlota (1991) *The Parameter of Aspect*, Kluwer.

Bibliography IV

Steedman, Mark (2005) The Productions of Time: Temporality and Causality in Linguistic Semantics. Unpublished lecture notes: `http://homepages.inf.ed.ac.uk/steedman/papers/temporality/temporality2.pdf`.

Verkuyl, Henk (1989) Aspectual Classes and Aspectual Composition, *Linguistics and Philosophy*, Vol. 12, pp. 39–94.