Extensions to Generative Lexicon

COSI 216
James Pustejovsky

Department of Computer Science
Brandeis University

October 9, 2009

Outline

Outline

Outline

Outline

Outline

Outline

Questions Addressed

- What conditions does a predicate impose on its arguments, and how are these conditions realized?
- How many meanings are needed for a word appearing in multiple syntactic contexts (i.e., polysemy)?
- What are the sources of polysemy?
- Given these facts, how can we maintain a compositional semantics?

Questions Addressed

- What conditions does a predicate impose on its arguments, and how are these conditions realized?
- How many meanings are needed for a word appearing in multiple syntactic contexts (i.e., polysemy)?
- What are the sources of polysemy?
- Given these facts, how can we maintain a compositional semantics?

Questions Addressed

- What conditions does a predicate impose on its arguments, and how are these conditions realized?
- How many meanings are needed for a word appearing in multiple syntactic contexts (i.e., polysemy)?
- What are the sources of polysemy?
- Given these facts, how can we maintain a compositional semantics?

Questions Addressed

- What conditions does a predicate impose on its arguments, and how are these conditions realized?
- How many meanings are needed for a word appearing in multiple syntactic contexts (i.e., polysemy)?
- What are the sources of polysemy?
- Given these facts, how can we maintain a compositional semantics?

Questions Addressed

- What conditions does a predicate impose on its arguments, and how are these conditions realized?
- How many meanings are needed for a word appearing in multiple syntactic contexts (i.e., polysemy)?
- What are the sources of polysemy?
- Given these facts, how can we maintain a compositional semantics?

Two Types of Polysemy

- Inherent polysemy: where multiple interpretations of an expression are available by virtue of the semantics inherent in the expression itself.
- selectional polysemy: where any novel interpretation of an expression is available due to contextual influences, namely, the type of the selecting expression.

1. a. John bought the new Obama book.
   b. John doesn't agree with the new Obama book.
   (inherent)
2. a. Mary left after her cigarette. (selectional)
   b. Mary left after her smoking a cigarette.

Two Types of Polysemy

- Inherent polysemy: where multiple interpretations of an expression are available by virtue of the semantics inherent in the expression itself.
- selectional polysemy: where any novel interpretation of an expression is available due to contextual influences, namely, the type of the selecting expression.

1. a. John bought the new Obama book.
   b. John doesn't agree with the new Obama book.
   (inherent)
2. a. Mary left after her cigarette. (selectional)
   b. Mary left after her smoking a cigarette.

Two Types of Polysemy

- Inherent polysemy: where multiple interpretations of an expression are available by virtue of the semantics inherent in the expression itself.
- selectional polysemy: where any novel interpretation of an expression is available due to contextual influences, namely, the type of the selecting expression.

1. a. John bought the new Obama book.
   b. John doesn't agree with the new Obama book.
   (inherent)
2. a. Mary left after her cigarette. (selectional)
   b. Mary left after her smoking a cigarette.

Two Types of Polysemy

- Inherent polysemy: where multiple interpretations of an expression are available by virtue of the semantics inherent in the expression itself.
- selectional polysemy: where any novel interpretation of an expression is available due to contextual influences, namely, the type of the selecting expression.

1. a. John bought the new Obama book.
   b. John doesn't agree with the new Obama book.
   (inherent)
2. a. Mary left after her cigarette. (selectional)
   b. Mary left after her smoking a cigarette.

Two Types of Polysemy

- Inherent polysemy: where multiple interpretations of an expression are available by virtue of the semantics inherent in the expression itself.
- selectional polysemy: where any novel interpretation of an expression is available due to contextual influences, namely, the type of the selecting expression.

1. a. John bought the new Obama book.
   b. John doesn't agree with the new Obama book.
   (inherent)
2. a. Mary left after her cigarette. (selectional)
   b. Mary left after her smoking a cigarette.

Systematic (Logical) Polysemy

1. There's chicken in the salad.
2. We'll have a water and two beers.
3. Roser finished her thesis.
4. Mary began the novel.
5. Mary believes John's story.
6. Mary believes John.

Systematic (Logical) Polysemy

1. There's chicken in the salad.
2. We'll have a water and two beers.
3. Roser finished her thesis.
4. Mary began the novel.
5. Mary believes John's story.
6. Mary believes John.

Systematic (Logical) Polysemy

1. There's chicken in the salad.
2. We'll have a water and two beers.
3. Roser finished her thesis.
4. Mary began the novel.
5. Mary believes John's story.
6. Mary believes John.

Systematic (Logical) Polysemy

1. There's chicken in the salad.
2. We'll have a water and two beers.
3. Roser finished her thesis.
4. Mary began the novel.
5. Mary believes John's story.
6. Mary believes John.

Systematic (Logical) Polysemy

1. There's chicken in the salad.
2. We'll have a water and two beers.
3. Roser finished her thesis.
4. Mary began the novel.
5. Mary believes John's story.
6. Mary believes John.

Systematic (Logical) Polysemy

1. There's chicken in the salad.
2. We'll have a water and two beers.
3. Roser finished her thesis.
4. Mary began the novel.
5. Mary believes John's story.
6. Mary believes John.

Flexibility of Subject Interpretation

Subject of kill:

- John killed Mary.
- The gun killed Mary.
- The shot killed Mary.
- The bullet killed Mary.
- John's pulling the trigger killed Mary.
- *The trigger killed Mary.

Flexibility of Subject Interpretation

Subject of kill:

- John killed Mary.
- The gun killed Mary.
- The shot killed Mary.
- The bullet killed Mary.
- John's pulling the trigger killed Mary.
- *The trigger killed Mary.

Flexibility of Subject Interpretation

Subject of kill:

- John killed Mary.
- The gun killed Mary.
- The shot killed Mary.
- The bullet killed Mary.
- John's pulling the trigger killed Mary.
- *The trigger killed Mary.

Flexibility of Subject Interpretation

Subject of kill:

- John killed Mary.
- The gun killed Mary.
- The shot killed Mary.
- The bullet killed Mary.
- John's pulling the trigger killed Mary.
- *The trigger killed Mary.

Flexibility of Subject Interpretation

Subject of kill:

- John killed Mary.
- The gun killed Mary.
- The shot killed Mary.
- The bullet killed Mary.
- John's pulling the trigger killed Mary.
- *The trigger killed Mary.

Flexibility of Subject Interpretation

Subject of kill:

- John killed Mary.
- The gun killed Mary.
- The shot killed Mary.
- The bullet killed Mary.
- John's pulling the trigger killed Mary.
- *The trigger killed Mary.

Flexibility of Subject Interpretation

Subject of kill:

- John killed Mary.
- The gun killed Mary.
- The shot killed Mary.
- The bullet killed Mary.
- John's pulling the trigger killed Mary.
- *The trigger killed Mary.

Causation and Intention

- John rolled down the hill as fast as he could.
- John cooled off with an iced latte.

Subject Rule (Wechsler, 2005): Optionally interpret subject as AGENTIVE.

kill vs murder:

- John killed the flowers accidently / intentionally.
- John murdered Mary.
- *John murdered Mary intentionally / accidentally.

Causation and Intention

- John rolled down the hill as fast as he could.
- John cooled off with an iced latte.

  Subject Rule (Wechsler, 2005): Optionally interpret subject as AGENTIVE.

  kill vs murder:

- John killed the flowers accidently / intentionally.
- John murdered Mary.
- *John murdered Mary intentionally / accidentally.

Causation and Intention

- John rolled down the hill as fast as he could.
- John cooled off with an iced latte.

  Subject Rule (Wechsler, 2005): Optionally interpret subject as AGENTIVE.

  kill vs murder:

- John killed the flowers accidently / intentionally.
- John murdered Mary.
- *John murdered Mary intentionally / accidentally.

Causation and Intention

- John rolled down the hill as fast as he could.
- John cooled off with an iced latte.

  Subject Rule (Wechsler, 2005): Optionally interpret subject as AGENTIVE.

  kill vs murder:

- John killed the flowers accidently / intentionally.
- John murdered Mary.
- *John murdered Mary intentionally / accidentally.

Causation and Intention

- John rolled down the hill as fast as he could.
- John cooled off with an iced latte.

Subject Rule (Wechsler, 2005): Optionally interpret subject as AGENTIVE.

kill vs murder:

- John killed the flowers accidently / intentionally.
- John murdered Mary.
- *John murdered Mary intentionally / accidentally.

Causation and Intention

- John rolled down the hill as fast as he could.
- John cooled off with an iced latte.

  Subject Rule (Wechsler, 2005): Optionally interpret subject as AGENTIVE.

  kill vs murder:

- John killed the flowers accidently / intentionally.
- John murdered Mary.
- *John murdered Mary intentionally / accidentally.

Causation and Intention

- John rolled down the hill as fast as he could.
- John cooled off with an iced latte.

  Subject Rule (Wechsler, 2005): Optionally interpret subject as AGENTIVE.

  kill vs murder:
- John killed the flowers accidently / intentionally.
- John murdered Mary.
- *John murdered Mary intentionally / accidentally.

Causation and Intention

- John rolled down the hill as fast as he could.
- John cooled off with an iced latte.

  Subject Rule (Wechsler, 2005): Optionally interpret subject as AGENTIVE.

  kill vs murder:

- John killed the flowers accidently / intentionally.
- John murdered Mary.
- *John murdered Mary intentionally / accidentally.

Flexibility of Object Interpretation

Fillmore (1985), Levin (1993), Levin and Rappaport (1998), Jackendoff (1990), Pustejovsky and Busa (1995)

- John swept [the dirt]$_{material}$.
- John swept [the room]$_{region}$.

- The man shoveled [the snow]$_{material}$.
- The man shoveled [the driveway]$_{region}$.

Flexibility of Object Interpretation

Fillmore (1985), Levin (1993), Levin and Rappaport (1998), Jackendoff (1990), Pustejovsky and Busa (1995)

- John swept [the dirt]$_{material}$.
- John swept [the room]$_{region}$.

- The man shoveled [the snow]$_{material}$.
- The man shoveled [the driveway]$_{region}$.

Flexibility of Object Interpretation

Fillmore (1985), Levin (1993), Levin and Rappaport (1998), Jackendoff (1990), Pustejovsky and Busa (1995)

- John swept [the dirt]$_{material}$.
- John swept [the room]$_{region}$.

- The man shoveled [the snow]$_{material}$.
- The man shoveled [the driveway]$_{region}$.

Flexibility of Object Interpretation

Fillmore (1985), Levin (1993), Levin and Rappaport (1998), Jackendoff (1990), Pustejovsky and Busa (1995)

- John swept [the dirt]$_{material}$.
- John swept [the room]$_{region}$.

- The man shoveled [the snow]$_{material}$.
- The man shoveled [the driveway]$_{region}$.

Flexibility of Object Interpretation

Fillmore (1985), Levin (1993), Levin and Rappaport (1998), Jackendoff (1990), Pustejovsky and Busa (1995)

- John swept [the dirt]$_{material}$.
- John swept [the room]$_{region}$.

- The man shoveled [the snow]$_{material}$.
- The man shoveled [the driveway]$_{region}$.

Flexibility of Arguments: Experiencers

1. That book bored me terribly.
2. The movie frightened Mary.
3. The newspaper article angered the Republicans.
4. Listening to Mary irritates Alice.

Flexibility of Arguments: Experiencers

1. That book bored me terribly.
2. The movie frightened Mary.
3. The newspaper article angered the Republicans.
4. Listening to Mary irritates Alice.

Flexibility of Arguments: Experiencers

1. That book bored me terribly.
2. The movie frightened Mary.
3. The newspaper article angered the Republicans.
4. Listening to Mary irritates Alice.

Flexibility of Arguments: Experiencers

1. That book bored me terribly.
2. The movie frightened Mary.
3. The newspaper article angered the Republicans.
4. Listening to Mary irritates Alice.

Flexibility of Arguments: Experiencers

1. That book bored me terribly.
2. The movie frightened Mary.
3. The newspaper article angered the Republicans.
4. Listening to Mary irritates Alice.

Flexibility of Arguments: Perception

- The boy heard a cat / a dog.
- They heard a bang / cry / rumor / shout / rain.
- !John heard the cloud/star/light.
- The crowd listened to the poem/speaker/speech.

Flexibility of Arguments: Perception

- The boy heard a cat / a dog.
- They heard a bang / cry / rumor / shout / rain.
- !John heard the cloud/star/light.
- The crowd listened to the poem/speaker/speech.

Flexibility of Arguments: Perception

- The boy heard a cat / a dog.
- They heard a bang / cry / rumor / shout / rain.
- !John heard the cloud/star/light.
- The crowd listened to the poem/speaker/speech.

Flexibility of Arguments: Perception

- The boy heard a cat / a dog.
- They heard a bang / cry / rumor / shout / rain.
- !John heard the cloud/star/light.
- The crowd listened to the poem/speaker/speech.

Flexibility of Arguments: Perception

- The boy heard a cat / a dog.
- They heard a bang / cry / rumor / shout / rain.
- !John heard the cloud/star/light.
- The crowd listened to the poem/speaker/speech.

Flexibility of Arguments: Attitudes, Factives

- Mary believes the rumor.
- No one believes the newspaper.
- She found the book hard to believe.
- They denied the actual conditions of the prisons.
- The graduate student regrets his last homework assignment.
- The hacker acknowledged the spam.

Flexibility of Arguments: Attitudes, Factives

- Mary believes the rumor.
- No one believes the newspaper.
- She found the book hard to believe.
- They denied the actual conditions of the prisons.
- The graduate student regrets his last homework assignment.
- The hacker acknowledged the spam.

Flexibility of Arguments: Attitudes, Factives

- Mary believes the rumor.
- No one believes the newspaper.
- She found the book hard to believe.
- They denied the actual conditions of the prisons.
- The graduate student regrets his last homework assignment.
- The hacker acknowledged the spam.

Flexibility of Arguments: Attitudes, Factives

- Mary believes the rumor.
- No one believes the newspaper.
- She found the book hard to believe.
- They denied the actual conditions of the prisons.
- The graduate student regrets his last homework assignment.
- The hacker acknowledged the spam.

Flexibility of Arguments: Attitudes, Factives

- Mary believes the rumor.
- No one believes the newspaper.
- She found the book hard to believe.
- They denied the actual conditions of the prisons.
- The graduate student regrets his last homework assignment.
- The hacker acknowledged the spam.

Flexibility of Arguments: Attitudes, Factives

- Mary believes the rumor.
- No one believes the newspaper.
- She found the book hard to believe.
- They denied the actual conditions of the prisons.
- The graduate student regrets his last homework assignment.
- The hacker acknowledged the spam.

Flexibility of Arguments: Attitudes, Factives

- Mary believes the rumor.
- No one believes the newspaper.
- She found the book hard to believe.
- They denied the actual conditions of the prisons.
- The graduate student regrets his last homework assignment.
- The hacker acknowledged the spam.

Flexibility of Arguments: Aspectuals

The verb begin is syntactically polymorphic:

- Mary began [to eat her breakfast].
- Mary began [eating her breakfast].
- Mary began [her breakfast].

but semantically underspecified:

- Mary began her beer/thesis/dinner/class/homework/bath
- John enjoyed his coffee/movie/cigar/discussion/appointment

Flexibility of Arguments: Aspectuals

The verb begin is syntactically polymorphic:

- Mary began [to eat her breakfast].
- Mary began [eating her breakfast].
- Mary began [her breakfast].

but semantically underspecified:

- Mary began her beer/thesis/dinner/class/homework/bath
- John enjoyed his coffee/movie/cigar/discussion/appointment

Flexibility of Arguments: Aspectuals

The verb begin is syntactically polymorphic:

- Mary began [to eat her breakfast].
- Mary began [eating her breakfast].
- Mary began [her breakfast].

but semantically underspecified:

- Mary began her beer/thesis/dinner/class/homework/bath
- John enjoyed his coffee/movie/cigar/discussion/appointment

Flexibility of Arguments: Aspectuals

The verb begin is syntactically polymorphic:

- Mary began [to eat her breakfast].
- Mary began [eating her breakfast].
- Mary began [her breakfast].

but semantically underspecified:

- Mary began
  her beer/thesis/dinner/class/homework/bath
- John enjoyed
  his coffee/movie/cigar/discussion/appointment

Flexibility of Arguments: Concealed Questions

- John knows [that the earth is round].
- John told Mary [that she is an idiot].
- Mary realizes [that she is mistaken].

- Mary knows [what time it is].
- John knows [how old she is].
- Mary told John [where she lives].
- John told me [how old he is].

- Mary knows the time.
- John knows her age.
- Mary told John her address.
- John told me his age.

Flexibility of Arguments: Concealed Questions

- John knows [that the earth is round].
- John told Mary [that she is an idiot].
- Mary realizes [that she is mistaken].

- Mary knows [what time it is].
- John knows [how old she is].
- Mary told John [where she lives].
- John told me [how old he is].

- Mary knows the time.
- John knows her age.
- Mary told John her address.
- John told me his age.

Flexibility of Arguments: Concealed Questions

- John knows [that the earth is round].
- John told Mary [that she is an idiot].
- Mary realizes [that she is mistaken].

- Mary knows [what time it is].
- John knows [how old she is].
- Mary told John [where she lives].
- John told me [how old he is].

- Mary knows the time.
- John knows her age.
- Mary told John her address.
- John told me his age.

Selection in a Compositional Theory

1. What elements can select?
2. What is an argument?
3. What does it mean for a predicate to select an argument?
4. How does selection relate to composition and lexical decomposition?

Verb Meaning

(1) a. Verb: V How do we decompose the meaning?
b. Arguments: x, y, z, ...

(2) a. Body: the predicate, with bound variables.
b. Arguments: the parameter list.

$$\overbrace{\lambda x_i}^{Args}\ \overbrace{[\Phi]}^{Body}$$

Verb Meaning

(1) a. Verb: V How do we decompose the meaning?
b. Arguments: x, y, z, ...

(2) a. Body: the predicate, with bound variables.
b. Arguments: the parameter list.

$$\overbrace{\lambda x_i}^{Args}\ \overbrace{[\Phi]}^{Body}$$

Verb Meaning

(1) a. Verb: V How do we decompose the meaning?
b. Arguments: x, y, z, ...

(2) a. Body: the predicate, with bound variables.
b. Arguments: the parameter list.

$$\overbrace{\lambda x_i}^{Args}\;\overbrace{[\Phi]}^{Body}$$

Verb Meaning

(1) a. Verb: V How do we decompose the meaning?
b. Arguments: x, y, z, ...

(2) a. Body: the predicate, with bound variables.
b. Arguments: the parameter list.

$$\overbrace{\lambda x_i}^{Args} \overbrace{[\Phi]}^{Body}$$

Verb Meaning

(1) a. Verb: V How do we decompose the meaning?
 b. Arguments: x, y, z, ...

(2) a. Body: the predicate, with bound variables.
 b. Arguments: the parameter list.

$$\overbrace{\lambda x_i}^{Args}\ \overbrace{[\Phi]}^{Body}$$

Verb Meaning

(1) a. Verb: V How do we decompose the meaning?
b. Arguments: x, y, z, ...

(2) a. Body: the predicate, with bound variables.
b. Arguments: the parameter list.

$$\overbrace{\lambda x_i}^{Args}\ \overbrace{[\Phi]}^{Body}$$

Decomposition Strategies

1. atomic predication: do nothing, $P(x_1)$
2. add arguments: $P(x_1) \Longrightarrow P(x_1, x_2)$
3. split the predicate: $P \Longrightarrow P_1, P_2$
4. add and split: $P(x_1) \Longrightarrow P(x_1, x_2), P_2(x_2)$

Decomposition Strategies

1. atomic predication: do nothing, $P(x_1)$
2. add arguments: $P(x_1) \Longrightarrow P(x_1, x_2)$
3. split the predicate: $P \Longrightarrow P_1, P_2$
4. add and split: $P(x_1) \Longrightarrow P(x_1, x_2), P_2(x_2)$

Decomposition Strategies

1. atomic predication: do nothing, $P(x_1)$
2. add arguments: $P(x_1) \Longrightarrow P(x_1, x_2)$
3. split the predicate: $P \Longrightarrow P_1, P_2$
4. add and split: $P(x_1) \Longrightarrow P(x_1, x_2), P_2(x_2)$

Decomposition Strategies

1. atomic predication: do nothing, $P(x_1)$
2. add arguments: $P(x_1) \Longrightarrow P(x_1, x_2)$
3. split the predicate: $P \Longrightarrow P_1, P_2$
4. add and split: $P(x_1) \Longrightarrow P(x_1, x_2), P_2(x_2)$

Decomposition Strategies

1. atomic predication: do nothing, $P(x_1)$
2. add arguments: $P(x_1) \Longrightarrow P(x_1, x_2)$
3. split the predicate: $P \Longrightarrow P_1, P_2$
4. add and split: $P(x_1) \Longrightarrow P(x_1, x_2), P_2(x_2)$

Atomic Predication

Syntax mirrors argument structure:

$$\text{Verb}(\text{Arg}_1, \ldots, \text{Arg}_n) \iff \lambda x_n \ldots \lambda x_1[\Phi]$$

1. $\lambda x[\text{die}(x)]$
   The flower died.
2. $\lambda y \lambda x[\text{hit}(x, y)]$
   The car hit the wall.

Atomic Predication

Syntax mirrors argument structure:

$$\text{Verb}(\text{Arg}_1, \ldots, \text{Arg}_n) \iff \lambda x_n \ldots \lambda x_1[\Phi]$$

1. $\lambda x[\text{die}(x)]$
   The flower died.
2. $\lambda y \lambda x[\text{hit}(x, y)]$
   The car hit the wall.

Atomic Predication

Syntax mirrors argument structure:

$\text{Verb}(\text{Arg}_1, \ldots, \text{Arg}_n) \iff \lambda x_n \ldots \lambda x_1[\Phi]$

1. $\lambda x[\text{die}(x)]$
   The flower died.
2. $\lambda y \lambda x[\text{hit}(x, y)]$
   The car hit the wall.

Atomic Predication

Syntax mirrors argument structure:

$$\text{Verb}(\text{Arg}_1, \ldots, \text{Arg}_n) \iff \lambda x_n \ldots \lambda x_1 [\Phi]$$

1. $\lambda x[\text{die}(x)]$
   The flower died.
2. $\lambda y \lambda x[\text{hit}(x, y)]$
   The car hit the wall.

Add Arguments

Parameter structure adds additional arguments for interpretation in the model:

$$\lambda x_m \ldots \lambda x_{n+1} \lambda x_n \ldots \lambda x_1 [\Phi] \Longrightarrow \text{Verb}(\text{Arg}_1, \ldots, \text{Arg}_n)$$

1. $\lambda y \lambda x \lambda e[\text{kill}(e, x, y)]$: (Davidson, 1967)
   The gardener killed the flower.
2. $\lambda l_2 \lambda l_1 \lambda x \lambda e[\text{go}(e, x, l_1, l_2)]$: (Hobbs, 1993)
   Nicholas went to China.
3. $\lambda t_2 \lambda t_1 \lambda l \lambda y \lambda x[\text{teach}(x, y, t_1, t_2, l)]$: (TimeML'07)
   Graham taught for an hour in Boston.

Add Arguments

Parameter structure adds additional arguments for interpretation in the model:

$$\lambda x_m \ldots \lambda x_{n+1} \lambda x_n \ldots \lambda x_1 [\Phi] \Longrightarrow \mathrm{Verb}(\mathrm{Arg}_1, \ldots, \mathrm{Arg}_n)$$

1. $\lambda y \lambda x \lambda e[\mathrm{kill}(e, x, y)]$: (Davidson, 1967)
   The gardener killed the flower.
2. $\lambda l_2 \lambda l_1 \lambda x \lambda e[\mathrm{go}(e, x, l_1, l_2)]$: (Hobbs, 1993)
   Nicholas went to China.
3. $\lambda t_2 \lambda t_1 \lambda l \lambda y \lambda x[\mathrm{teach}(x, y, t_1, t_2, l)]$: (TimeML'07)
   Graham taught for an hour in Boston.

Add Arguments

Parameter structure adds additional arguments for interpretation in the model:

$$\lambda x_m \ldots \lambda x_{n+1} \lambda x_n \ldots \lambda x_1 [\Phi] \Longrightarrow \text{Verb}(\text{Arg}_1, \ldots, \text{Arg}_n)$$

1. $\lambda y \lambda x \lambda e[\text{kill}(e, x, y)]$: (Davidson, 1967)
   The gardener killed the flower.
2. $\lambda l_2 \lambda l_1 \lambda x \lambda e[\text{go}(e, x, l_1, l_2)]$: (Hobbs, 1993)
   Nicholas went to China.
3. $\lambda t_2 \lambda t_1 \lambda l \lambda y \lambda x[\text{teach}(x, y, t_1, t_2, l)]$: (TimeML'07)
   Graham taught for an hour in Boston.

Add Arguments

Parameter structure adds additional arguments for interpretation in the model:

$$\lambda x_m \ldots \lambda x_{n+1} \lambda x_n \ldots \lambda x_1 [\Phi] \Longrightarrow \text{Verb}(\text{Arg}_1, \ldots, \text{Arg}_n)$$

1. $\lambda y \lambda x \lambda e[\text{kill}(e, x, y)]$: (Davidson, 1967)
   The gardener killed the flower.
2. $\lambda l_2 \lambda l_1 \lambda x \lambda e[\text{go}(e, x, l_1, l_2)]$: (Hobbs, 1993)
   Nicholas went to China.
3. $\lambda t_2 \lambda t_1 \lambda l \lambda y \lambda x[\text{teach}(x, y, t_1, t_2, l)]$: (TimeML'07)
   Graham taught for an hour in Boston.

Add Arguments

Parameter structure adds additional arguments for interpretation in the model:

$$\lambda x_m \ldots \lambda x_{n+1} \lambda x_n \ldots \lambda x_1 [\Phi] \Longrightarrow \mathrm{Verb}(\mathrm{Arg}_1, \ldots, \mathrm{Arg}_n)$$

1. $\lambda y \lambda x \lambda e[\mathrm{kill}(e, x, y)]$: (Davidson, 1967)
   The gardener killed the flower.
2. $\lambda l_2 \lambda l_1 \lambda x \lambda e[\mathrm{go}(e, x, l_1, l_2)]$: (Hobbs, 1993)
   Nicholas went to China.
3. $\lambda t_2 \lambda t_1 \lambda l \lambda y \lambda x[\mathrm{teach}(x, y, t_1, t_2, l)]$: (TimeML'07)
   Graham taught for an hour in Boston.

Add Arguments

Parameter structure adds additional arguments for interpretation in the model:

$$\lambda x_m \ldots \lambda x_{n+1} \lambda x_n \ldots \lambda x_1 [\Phi] \Longrightarrow \mathrm{Verb}(\mathrm{Arg}_1, \ldots, \mathrm{Arg}_n)$$

1. $\lambda y \lambda x \lambda e[\mathrm{kill}(e, x, y)]$: (Davidson, 1967)
   The gardener killed the flower.
2. $\lambda l_2 \lambda l_1 \lambda x \lambda e[\mathrm{go}(e, x, l_1, l_2)]$: (Hobbs, 1993)
   Nicholas went to China.
3. $\lambda t_2 \lambda t_1 \lambda l \lambda y \lambda x[\mathrm{teach}(x, y, t_1, t_2, l)]$: (TimeML'07)
   Graham taught for an hour in Boston.

Split The Predicate

P is defined as a complex expression of subpredicates over the parameter:

$$\text{Verb}(\text{Arg}_1) \Longrightarrow \lambda x[\Phi_1, \ldots \Phi_k]$$

1. die: $\lambda x[\text{alive}(x) \wedge \text{Become}(\neg\text{alive}(x))]$
   The flower died.
2. bachelor: $\lambda x[\text{male}(x) \wedge \text{person}(x) \wedge \text{adult}(x) \wedge \neg\text{married}(x)]$

Split The Predicate

P is defined as a complex expression of subpredicates over the parameter:

$$\text{Verb}(\text{Arg}_1) \Longrightarrow \lambda x[\Phi_1, \ldots \Phi_k]$$

1. die: $\lambda x[\text{alive}(x) \wedge \text{Become}(\neg\text{alive}(x))]$
   The flower died.
2. bachelor: $\lambda x[\text{male}(x) \wedge \text{person}(x) \wedge \text{adult}(x) \wedge \neg\text{married}(x)]$

Split The Predicate

P is defined as a complex expression of subpredicates over the parameter:

$$\text{Verb}(\text{Arg}_1) \Longrightarrow \lambda x[\Phi_1, \ldots \Phi_k]$$

1. die: $\lambda x[\text{alive}(x) \wedge \text{Become}(\neg\text{alive}(x))]$
   The flower died.
2. bachelor: $\lambda x[\text{male}(x) \wedge \text{person}(x) \wedge \text{adult}(x) \wedge \neg\text{married}(x)]$

Split The Predicate

P is defined as a complex expression of subpredicates over the parameter:

$$\text{Verb}(\text{Arg}_1) \Longrightarrow \lambda x[\Phi_1, \ldots \Phi_k]$$

1. die: $\lambda x[\text{alive}(x) \wedge \text{Become}(\neg\text{alive}(x))]$
   The flower died.
2. bachelor: $\lambda x[\text{male}(x) \wedge \text{person}(x) \wedge \text{adult}(x) \wedge \neg\text{married}(x)]$

Split The Predicate

P is defined as a complex expression of subpredicates over the parameter:

$$\text{Verb}(\text{Arg}_1) \Longrightarrow \lambda x[\Phi_1, \ldots \Phi_k]$$

1. die: $\lambda x[\text{alive}(x) \wedge \text{Become}(\neg\text{alive}(x))]$
   The flower died.
2. bachelor: $\lambda x[\text{male}(x) \wedge \text{person}(x) \wedge \text{adult}(x) \wedge \neg\text{married}(x)]$

Add and Split

Parameter structure is enhanced, and P is defined as a complex of subpredicates:

$$\text{Verb}(\text{Arg}_1, \ldots, \text{Arg}_n) \implies \lambda x_m \ldots \lambda x_{n+1} \lambda x_n \ldots \lambda x_1 [\Phi_1, \ldots \Phi_k]$$

1. kill:
$\lambda yxe_1 e_2[act(e_1, x, y) \wedge \neg dead(e_1, y) \wedge dead(e_2, x) \wedge e_1 < e_2]$:
The gardner killed the flower.

Add and Split

Parameter structure is enhanced, and P is defined as a complex of subpredicates:

$$\text{Verb}(\text{Arg}_1, \ldots, \text{Arg}_n) \implies$$
$$\lambda x_m \ldots \lambda x_{n+1} \lambda x_n \ldots \lambda x_1 [\Phi_1, \ldots \Phi_k]$$

1. kill:
$\lambda y x e_1 e_2 [\text{act}(e_1, x, y) \wedge \neg \text{dead}(e_1, y) \wedge \text{dead}(e_2, x) \wedge e_1 < e_2]$:
The gardner killed the flower.

Add and Split

Parameter structure is enhanced, and P is defined as a complex of subpredicates:

$$\text{Verb}(\text{Arg}_1, \ldots, \text{Arg}_n) \quad \Longrightarrow$$
$$\lambda x_m \ldots \lambda x_{n+1} \lambda x_n \ldots \lambda x_1 [\Phi_1, \ldots \Phi_k]$$

1. kill:
   $\lambda y x e_1 e_2 [\text{act}(e_1, x, y) \wedge \neg \text{dead}(e_1, y) \wedge \text{dead}(e_2, x) \wedge e_1 < e_2]$:
   The gardner killed the flower.

Add and Split

Parameter structure is enhanced, and P is defined as a complex of subpredicates:

$\mathrm{Verb}(\mathrm{Arg}_1, \ldots, \mathrm{Arg}_n) \implies$
$\lambda x_m \ldots \lambda x_{n+1} \lambda x_n \ldots \lambda x_1 [\Phi_1, \ldots \Phi_k]$

1. kill:
$\lambda y x e_1 e_2 [\mathrm{act}(e_1, x, y) \wedge \neg \mathrm{dead}(e_1, y) \wedge \mathrm{dead}(e_2, x) \wedge e_1 < e_2]$:
The gardner killed the flower.

Supralexical Composition: Kratzer (1996,2002)

Parameter structure is enriched through mechanism of additional operators, while *P* is enriched by an additional operation:

- $\text{Verb}(\text{Arg}_1, \ldots, \text{Arg}_n) \Longrightarrow \lambda x_n \ldots \lambda x_1[\Phi]$
- $v \Longrightarrow \lambda f_\sigma \lambda x_1[\mathcal{R}(f)(x_1)]$
- $\Longrightarrow \lambda f_\sigma \lambda x_1[\mathcal{R}(f)(x_1)](\lambda x[\Phi])_\sigma$
- $\Longrightarrow \lambda x_1[\mathcal{R}([\Phi])(x_1)]$
- Event Identification: adds an argument through composition

Supralexical Composition: Kratzer (1996,2002)

Parameter structure is enriched through mechanism of additional operators, while P is enriched by an additional operation:

- $\text{Verb}(\text{Arg}_1, \ldots, \text{Arg}_n) \Longrightarrow \lambda x_n \ldots \lambda x_1[\Phi]$
- $v \Longrightarrow \lambda f_\sigma \lambda x_1[\mathcal{R}(f)(x_1)]$
- $\Longrightarrow \lambda f_\sigma \lambda x_1[\mathcal{R}(f)(x_1)](\lambda x[\Phi])_\sigma$
- $\Longrightarrow \lambda x_1[\mathcal{R}([\Phi])(x_1)]$
- Event Identification: adds an argument through composition

Supralexical Composition: Kratzer (1996,2002)

Parameter structure is enriched through mechanism of additional operators, while P is enriched by an additional operation:

- $\text{Verb}(\text{Arg}_1, \ldots, \text{Arg}_n) \Longrightarrow \lambda x_n \ldots \lambda x_1 [\Phi]$
- $v \Longrightarrow \lambda f_\sigma \lambda x_1 [\mathcal{R}(f)(x_1)]$
- $\Longrightarrow \lambda f_\sigma \lambda x_1 [\mathcal{R}(f)(x_1)](\lambda x[\Phi])_\sigma$
- $\Longrightarrow \lambda x_1 [\mathcal{R}([\Phi])(x_1)]$
- Event Identification: adds an argument through composition

Supralexical Composition: Kratzer (1996,2002)

Parameter structure is enriched through mechanism of additional operators, while *P* is enriched by an additional operation:

- $\text{Verb}(\text{Arg}_1, \ldots, \text{Arg}_n) \Longrightarrow \lambda x_n \ldots \lambda x_1[\Phi]$
- $v \Longrightarrow \lambda f_\sigma \lambda x_1[\mathcal{R}(f)(x_1)]$
- $\Longrightarrow \lambda f_\sigma \lambda x_1[\mathcal{R}(f)(x_1)](\lambda x[\Phi])_\sigma$
- $\Longrightarrow \lambda x_1[\mathcal{R}([\Phi])(x_1)]$
- Event Identification: adds an argument through composition

Supralexical Composition: Kratzer (1996,2002)

Parameter structure is enriched through mechanism of additional operators, while *P* is enriched by an additional operation:

- $\text{Verb}(\text{Arg}_1, \ldots, \text{Arg}_n) \implies \lambda x_n \ldots \lambda x_1[\Phi]$
- $v \implies \lambda f_\sigma \lambda x_1[\mathcal{R}(f)(x_1)]$
- $\implies \lambda f_\sigma \lambda x_1[\mathcal{R}(f)(x_1)](\lambda x[\Phi])_\sigma$
- $\implies \lambda x_1[\mathcal{R}([\Phi])(x_1)]$
- Event Identification: adds an argument through composition

Supralexical Composition: Kratzer (1996,2002)

Parameter structure is enriched through mechanism of additional operators, while *P* is enriched by an additional operation:

- $\mathrm{Verb}(\mathrm{Arg}_1, \ldots, \mathrm{Arg}_n) \Longrightarrow \lambda x_n \ldots \lambda x_1 [\Phi]$
- $v \Longrightarrow \lambda f_\sigma \lambda x_1 [\mathcal{R}(f)(x_1)]$
- $\Longrightarrow \lambda f_\sigma \lambda x_1 [\mathcal{R}(f)(x_1)](\lambda x[\Phi])_\sigma$
- $\Longrightarrow \lambda x_1 [\mathcal{R}([\Phi])(x_1)]$
- Event Identification: adds an argument through composition

Argument Typing as Abstracting from the Predicate

Richer typing for arguments:

1. Identifies specific predicates in the body of the expression that are characteristic functions of an argument;
2. pulls this subset of predicates out of the body, and creates a *pretest* to the expression as a restricted quantification over a domain of sorts, denoted by that set of predicates.

Argument Typing as Abstracting from the Predicate

Richer typing for arguments:

1. Identifies specific predicates in the body of the expression that are characteristic functions of an argument;
2. pulls this subset of predicates out of the body, and creates a *pretest* to the expression as a restricted quantification over a domain of sorts, denoted by that set of predicates.

Argument Typing as Abstracting from the Predicate

Richer typing for arguments:

1. Identifies specific predicates in the body of the expression that are characteristic functions of an argument;
2. pulls this subset of predicates out of the body, and creates a *pretest* to the expression as a restricted quantification over a domain of sorts, denoted by that set of predicates.

Types from Predicative Content

$$\lambda x_2 \lambda x_1 [\Phi_1, \ldots \overbrace{\Phi_{x_1}}^{\tau}, \ldots \overbrace{\Phi_{x_2}}^{\sigma}, \ldots, \Phi_k]$$

$$\lambda x_2 : \sigma \; \lambda x_1 : \tau [\Phi_1, \ldots, \Phi_k - \{\Phi_{x_1}, \Phi_{x_2}\}]$$

σ and τ have now become reified as types on the arguments.

Types from Predicative Content

$$\lambda x_2 \lambda x_1 [\Phi_1, \ldots \overbrace{\Phi_{x_1}}^{\tau}, \ldots \overbrace{\Phi_{x_2}}^{\sigma}, \ldots, \Phi_k]$$

$$\lambda x_2 : \sigma \; \lambda x_1 : \tau [\Phi_1, \ldots, \Phi_k - \{\Phi_{x_1}, \Phi_{x_2}\}]$$

σ and τ have now become reified as types on the arguments.

Types from Predicative Content

$$\lambda x_2 \lambda x_1 [\Phi_1, \ldots \overbrace{\Phi_{x_1}}^{\tau}, \ldots \overbrace{\Phi_{x_2}}^{\sigma}, \ldots, \Phi_k]$$

$$\lambda x_2 : \sigma \; \lambda x_1 : \tau [\Phi_1, \ldots, \Phi_k - \{\Phi_{x_1}, \Phi_{x_2}\}]$$

σ and τ have now become reified as types on the arguments.

Types from Predicative Content

$$\lambda x_2 \lambda x_1 [\Phi_1, \ldots \overbrace{\Phi_{x_1}}^{\tau}, \ldots \overbrace{\Phi_{x_2}}^{\sigma}, \ldots, \Phi_k]$$

$$\lambda x_2 : \sigma \ \lambda x_1 : \tau [\Phi_1, \ldots, \Phi_k - \{\Phi_{x_1}, \Phi_{x_2}\}]$$

σ and τ have now become reified as types on the arguments.

A Flexible Strategy of Selection

Arguments can be viewed as encoding pretests for performing the action in the predicate.

If the argument condition (i.e., its type) is not satisfied, the predicate either:

- fails to be interpreted (strong selection);
- coerces its argument according to a given set of strategies.

A Flexible Strategy of Selection

Arguments can be viewed as encoding pretests for performing the action in the predicate.

If the argument condition (i.e., its type) is not satisfied, the predicate either:

- fails to be interpreted (strong selection);
- coerces its argument according to a given set of strategies.

A Flexible Strategy of Selection

Arguments can be viewed as encoding pretests for performing the action in the predicate.

If the argument condition (i.e., its type) is not satisfied, the predicate either:

- fails to be interpreted (strong selection);
- coerces its argument according to a given set of strategies.

A Flexible Strategy of Selection

Arguments can be viewed as encoding pretests for performing the action in the predicate.

> If the argument condition (i.e., its type) is not satisfied, the predicate either:
>
> - fails to be interpreted (strong selection);
> - coerces its argument according to a given set of strategies.

Lexical Data Structures

(1) a. LEXICAL TYPING STRUCTURE: giving an explicit type for a word positioned within a type system for the language;
b. ARGUMENT STRUCTURE: specifying the number and nature of the arguments to a predicate;
c. EVENT STRUCTURE: defining the event type of the expression and any subeventual structure it may have;
d. QUALIA STRUCTURE: a structural differentiation of the predicative force for a lexical item.

Lexical Data Structures

(2) a. LEXICAL TYPING STRUCTURE: giving an explicit type for a word positioned within a type system for the language;
b. ARGUMENT STRUCTURE: specifying the number and nature of the arguments to a predicate;
c. EVENT STRUCTURE: defining the event type of the expression and any subeventual structure it may have;
d. QUALIA STRUCTURE: a structural differentiation of the predicative force for a lexical item.

Lexical Data Structures

(3) a. LEXICAL TYPING STRUCTURE: giving an explicit type for a word positioned within a type system for the language;
b. ARGUMENT STRUCTURE: specifying the number and nature of the arguments to a predicate;
c. EVENT STRUCTURE: defining the event type of the expression and any subeventual structure it may have;
d. QUALIA STRUCTURE: a structural differentiation of the predicative force for a lexical item.

Lexical Data Structures

(4) a. LEXICAL TYPING STRUCTURE: giving an explicit type for a word positioned within a type system for the language;
b. ARGUMENT STRUCTURE: specifying the number and nature of the arguments to a predicate;
c. EVENT STRUCTURE: defining the event type of the expression and any subeventual structure it may have;
d. QUALIA STRUCTURE: a structural differentiation of the predicative force for a lexical item.

Qualia

(5) a. FORMAL: the basic category of which distinguishes the meaning of a word within a larger domain;

b. CONSTITUTIVE: the relation between an object and its constituent parts;

c. TELIC: the purpose or function of the object, if there is one;

d. AGENTIVE: the factors involved in the object's origins or "coming into being".

Qualia

(6) a. FORMAL: the basic category of which distinguishes the meaning of a word within a larger domain;
b. CONSTITUTIVE: the relation between an object and its constituent parts;
c. TELIC: the purpose or function of the object, if there is one;
d. AGENTIVE: the factors involved in the object's origins or "coming into being".

Qualia

(7) a. FORMAL: the basic category of which distinguishes the meaning of a word within a larger domain;
b. CONSTITUTIVE: the relation between an object and its constituent parts;
c. TELIC: the purpose or function of the object, if there is one;
d. AGENTIVE: the factors involved in the object's origins or "coming into being".

Qualia

(8) a. FORMAL: the basic category of which distinguishes the meaning of a word within a larger domain;
b. CONSTITUTIVE: the relation between an object and its constituent parts;
c. TELIC: the purpose or function of the object, if there is one;
d. AGENTIVE: the factors involved in the object's origins or "coming into being".

Arguments and Body in GL

$$\overbrace{\overbrace{\lambda x_n \ldots \lambda x_1}^{AS}\ \overbrace{\lambda e_m \ldots \lambda e_1}^{ES}}^{Args}\ \overbrace{[Q_1 \wedge Q_2 \wedge Q_3 \wedge Q_4;\ C]}^{Body}$$

AS: Argument Structure

ES: Event Structure

Q_i: Qualia Structure

C: Constraints

GL Feature Structure

$$
\left[\begin{array}{l}
\alpha \\
\text{ARGSTR} = \left[\begin{array}{l} \text{ARG1} = x \\ \ldots \end{array}\right] \\
\text{EVENTSTR} = \left[\begin{array}{l} \text{EVENT1} = e1 \\ \text{EVENT2} = e2 \end{array}\right] \\
\text{QUALIA} = \left[\begin{array}{l} \text{CONST} = \textbf{what } x \textbf{ is made of} \\ \text{FORMAL} = \textbf{what } x \textbf{ is} \\ \text{TELIC} = e_2\textbf{: function of } x \\ \text{AGENTIVE} = e_1\textbf{: how } x \textbf{ came into being} \end{array}\right]
\end{array}\right]
$$

Type Composition Logic (Asher and Pustejovsky, 2006)

1. e the general type of entities; t the type of truth values. (σ, τ range over all simple types, and subtypes of e.)
2. If σ and τ are types, then so is $\sigma \to \tau$.
3. If σ and τ are types, then so is $\sigma \otimes_R \tau$; R ranges over A or T.
4. If σ and τ are types, then so is $\sigma \bullet \tau$.

Type Composition Logic (Asher and Pustejovsky, 2006)

1. e the general type of entities; t the type of truth values. (σ, τ range over all simple types, and subtypes of e.)
2. If σ and τ are types, then so is $\sigma \rightarrow \tau$.
3. If σ and τ are types, then so is $\sigma \otimes_R \tau$; R ranges over A or T.
4. If σ and τ are types, then so is $\sigma \bullet \tau$.

Type Composition Logic (Asher and Pustejovsky, 2006)

1. e the general type of entities; t the type of truth values. (σ, τ range over all simple types, and subtypes of e.)
2. If σ and τ are types, then so is $\sigma \to \tau$.
3. If σ and τ are types, then so is $\sigma \otimes_R \tau$; R ranges over A or T.
4. If σ and τ are types, then so is $\sigma \bullet \tau$.

Type Composition Logic (Asher and Pustejovsky, 2006)

1. e the general type of entities; t the type of truth values. (σ, τ range over all simple types, and subtypes of e.)
2. If σ and τ are types, then so is $\sigma \to \tau$.
3. If σ and τ are types, then so is $\sigma \otimes_R \tau$; R ranges over A or T.
4. If σ and τ are types, then so is $\sigma \bullet \tau$.

Type Composition Logic (Asher and Pustejovsky, 2006)

1. e the general type of entities; t the type of truth values. (σ, τ range over all simple types, and subtypes of e.)
2. If σ and τ are types, then so is $\sigma \rightarrow \tau$.
3. If σ and τ are types, then so is $\sigma \otimes_R \tau$; R ranges over A or T.
4. If σ and τ are types, then so is $\sigma \bullet \tau$.

Qualia Types

$$\begin{bmatrix} x: & & \alpha \\ & \otimes_c & \beta \\ & \otimes_t & \tau \\ & \otimes_a & \sigma \end{bmatrix}$$

with an unlabeled qualia value

$$\begin{bmatrix} x: & & \alpha \\ & \otimes & \tau \\ & & \end{bmatrix}$$

Qualia Types

$$\begin{bmatrix} x: & & \alpha \\ & \otimes_c & \beta \\ & \otimes_t & \tau \\ & \otimes_a & \sigma \end{bmatrix}$$

with an unlabeled qualia value

$$\begin{bmatrix} x: & & \alpha \\ & \otimes & \tau \\ & & \end{bmatrix}$$

Qualia Types

$$\begin{bmatrix} x: & \alpha \\ \bigotimes_c & \beta \\ \bigotimes_t & \tau \\ \bigotimes_a & \sigma \end{bmatrix}$$

with an unlabeled qualia value

$$\begin{bmatrix} x: & \alpha \\ \bigotimes & \tau \\ & \end{bmatrix}$$

Natural Types

Entities formed from the application of the FORMAL and/or CONST qualia roles:

1. For the predicates below, e_N is structured as a join semi-lattice, $\langle e_N, \sqsubseteq \rangle$;
2. *physical*, *human*, *stick*, *lion*, *pebble*
3. *water*, *sky*, *rock*

Natural Types

Entities formed from the application of the FORMAL and/or CONST qualia roles:

1. For the predicates below, e_N is structured as a join semi-lattice, $\langle e_N, \sqsubseteq \rangle$;
2. *physical, human, stick, lion, pebble*
3. *water, sky, rock*

Natural Types

Entities formed from the application of the FORMAL and/or CONST qualia roles:

1. For the predicates below, e_N is structured as a join semi-lattice, $\langle e_N, \sqsubseteq \rangle$;
2. *physical*, *human*, *stick*, *lion*, *pebble*
3. *water*, *sky*, *rock*

Natural Predicate Types

Predicates formed with Natural Entities as arguments:

1. *fall*: $e_N \rightarrow t$
2. *touch*: $e_N \rightarrow (e_N \rightarrow t)$
3. *be under*: $e_N \rightarrow (e_N \rightarrow t)$

a. $\lambda x : e_N[fall(x)]$

b. $\lambda y : e_N \lambda x : e_N[touch(x,y)]$

c. $\lambda y : e_N \lambda x : e_N[be\text{-}under(x,y)]$

Natural Predicate Types

Predicates formed with Natural Entities as arguments:

1. *fall*: $e_N \rightarrow t$
2. *touch*: $e_N \rightarrow (e_N \rightarrow t)$
3. *be under*: $e_N \rightarrow (e_N \rightarrow t)$

a. $\lambda x : e_N[\mathit{fall}(x)]$

b. $\lambda y : e_N \lambda x : e_N[\mathit{touch}(x,y)]$

c. $\lambda y : e_N \lambda x : e_N[\mathit{be\text{-}under}(x,y)]$

Natural Predicate Types

Predicates formed with Natural Entities as arguments:

1. *fall*: $e_N \rightarrow t$
2. *touch*: $e_N \rightarrow (e_N \rightarrow t)$
3. *be under*: $e_N \rightarrow (e_N \rightarrow t)$

a. $\lambda x : e_N[\textit{fall}(x)]$

b. $\lambda y : e_N \lambda x : e_N[\textit{touch}(x,y)]$

c. $\lambda y : e_N \lambda x : e_N[\textit{be-under}(x,y)]$

Natural Predicate Types

Predicates formed with Natural Entities as arguments:

1. *fall*: $e_N \rightarrow t$
2. *touch*: $e_N \rightarrow (e_N \rightarrow t)$
3. *be under*: $e_N \rightarrow (e_N \rightarrow t)$

a. $\lambda x : e_N[\mathit{fall}(\mathrm{x})]$

b. $\lambda y : e_N \lambda x : e_N[\mathit{touch}(\mathrm{x,y})]$

c. $\lambda y : e_N \lambda x : e_N[\mathit{be\text{-}under}(\mathrm{x,y})]$

Natural Predicate Types

Predicates formed with Natural Entities as arguments:

1. *fall*: $e_N \rightarrow t$
2. *touch*: $e_N \rightarrow (e_N \rightarrow t)$
3. *be under*: $e_N \rightarrow (e_N \rightarrow t)$

a. $\lambda x : e_N[\mathit{fall}(\mathrm{x})]$

b. $\lambda y : e_N \lambda x : e_N[\mathit{touch}(\mathrm{x,y})]$

c. $\lambda y : e_N \lambda x : e_N[\mathit{be\text{-}under}(\mathrm{x,y})]$

Natural Predicate Types

Predicates formed with Natural Entities as arguments:

1. *fall*: $e_N \rightarrow t$
2. *touch*: $e_N \rightarrow (e_N \rightarrow t)$
3. *be under*: $e_N \rightarrow (e_N \rightarrow t)$

a. $\lambda x : e_N[\textit{fall}(\mathrm{x})]$

b. $\lambda y : e_N \lambda x : e_N[\textit{touch}(\mathrm{x,y})]$

c. $\lambda y : e_N \lambda x : e_N[\textit{be-under}(\mathrm{x,y})]$

Natural Predicate Types

Predicates formed with Natural Entities as arguments:

1. *fall*: $e_N \rightarrow t$
2. *touch*: $e_N \rightarrow (e_N \rightarrow t)$
3. *be under*: $e_N \rightarrow (e_N \rightarrow t)$

a. $\lambda x : e_N[\textit{fall}(\mathrm{x})]$

b. $\lambda y : e_N \lambda x : e_N[\textit{touch}(\mathrm{x,y})]$

c. $\lambda y : e_N \lambda x : e_N[\textit{be-under}(\mathrm{x,y})]$

Artifactual Entity Types

Entities formed from the Naturals by adding the AGENTIVE or TELIC qualia roles:

1. Artifact Entity: $x : e_N \otimes_a \sigma$
   x exists because of event σ
2. Functional Entity: $x : e_N \otimes_t \tau$
   the purpose of x is τ
3. Functional Artifactual Entity: $x : (e_N \otimes_a \sigma) \otimes_t \tau$
   x exists because of event σ for the purpose τ

a. *beer*: (*liquid* $\otimes_a$ *brew*) $\otimes_t$ *drink*
b. *knife*: (*phys* $\otimes_a$ *make*) $\otimes_t$ *cut*
c. *house*: (*phys* $\otimes_a$ *build*) $\otimes_t$ *live_in*

Artifactual Predicate Types

Predicates formed with Artifactual Entities as arguments:

1. *spoil*: $e_N \otimes_t \tau \to t$
2. *fix*: $e_N \otimes_t \tau \to (e_N \to t)$

a. $\lambda x : e_A[\textit{spoil}(x)]$

b. $\lambda y : e_A \lambda x : e_N[\textit{fix}(x,y)]$

- The beer spoiled.
- Mary fixed the watch.

Artifactual Predicate Types

Predicates formed with Artifactual Entities as arguments:

1. *spoil*: $e_N \otimes_t \tau \rightarrow t$
2. *fix*: $e_N \otimes_t \tau \rightarrow (e_N \rightarrow t)$

a. $\lambda x: e_A[spoil(x)]$

b. $\lambda y: e_A \lambda x: e_N[fix(x,y)]$

- The beer spoiled.
- Mary fixed the watch.

Artifactual Predicate Types

Predicates formed with Artifactual Entities as arguments:

1. *spoil*: $e_N \otimes_t \tau \to t$
2. *fix*: $e_N \otimes_t \tau \to (e_N \to t)$

a. $\lambda x : e_A[spoil(x)]$

b. $\lambda y : e_A \lambda x : e_N[fix(x,y)]$

- The beer spoiled.
- Mary fixed the watch.

Artifactual Predicate Types

Predicates formed with Artifactual Entities as arguments:

1. *spoil*: $e_N \otimes_t \tau \rightarrow t$
2. *fix*: $e_N \otimes_t \tau \rightarrow (e_N \rightarrow t)$

a. $\lambda x : e_A[\mathit{spoil}(\mathrm{x})]$

b. $\lambda y : e_A \lambda x : e_N[\mathit{fix}(\mathrm{x,y})]$

- The beer spoiled.
- Mary fixed the watch.

Artifactual Predicate Types

Predicates formed with Artifactual Entities as arguments:

1. *spoil*: $e_N \otimes_t \tau \rightarrow t$
2. *fix*: $e_N \otimes_t \tau \rightarrow (e_N \rightarrow t)$

a. $\lambda x : e_A[\mathit{spoil}(\mathrm{x})]$

b. $\lambda y : e_A \lambda x : e_N[\mathit{fix}(\mathrm{x,y})]$

- The beer spoiled.
- Mary fixed the watch.

Artifactual Predicate Types

Predicates formed with Artifactual Entities as arguments:

1. *spoil*: $e_N \otimes_t \tau \rightarrow t$
2. *fix*: $e_N \otimes_t \tau \rightarrow (e_N \rightarrow t)$

a. $\lambda x: e_A[spoil(x)]$
b. $\lambda y: e_A \lambda x: e_N[fix(x,y)]$

- The beer spoiled.
- Mary fixed the watch.

Artifactual Predicate Types

Predicates formed with Artifactual Entities as arguments:

1. *spoil*: $e_N \otimes_t \tau \rightarrow t$
2. *fix*: $e_N \otimes_t \tau \rightarrow (e_N \rightarrow t)$

a. $\lambda x : e_A[spoil(\mathrm{x})]$

b. $\lambda y : e_A \lambda x : e_N[fix(\mathrm{x,y})]$

- The beer spoiled.
- Mary fixed the watch.

Complex Entity Types

Entities formed from the Naturals and Artifactuals by a product type between the entities, i.e., the dot, •.

1. a. Mary doesn't believe the book.
   b. John sold his book to Mary.
2. a. The exam started at noon.
   b. The students could not understand the exam.

Complex Entity Types

Entities formed from the Naturals and Artifactuals by a product type between the entities, i.e., the dot, •.

1. a. Mary doesn't believe the book.
   b. John sold his book to Mary.
2. a. The exam started at noon.
   b. The students could not understand the exam.

Complex Entity Types

Entities formed from the Naturals and Artifactuals by a product type between the entities, i.e., the dot, •.

1. a. Mary doesn't believe the book.
   b. John sold his book to Mary.
2. a. The exam started at noon.
   b. The students could not understand the exam.

Motivating Dot Objects

When a single word or phrase has the ability to appear in selected contexts that are contradictory in type specification.

If a lexical expression, α, where $\sigma \sqcap \tau = \bot$:

1. $[\_\_\_]_\sigma$ X
2. $[\_\_\_]_\tau$ Y

are both well-formed predications, then α is a dot object (complex type).

Motivating Dot Objects

When a single word or phrase has the ability to appear in selected contexts that are contradictory in type specification.

If a lexical expression, α, where $\sigma \sqcap \tau = \perp$:

1. $[\_\_\_]_{\sigma}$ X
2. $[\_\_\_]_{\tau}$ Y

are both well-formed predications, then α is a dot object (complex type).

Motivating Dot Objects

When a single word or phrase has the ability to appear in selected contexts that are contradictory in type specification.

If a lexical expression, α, where $\sigma \sqcap \tau = \perp$:

1. $[\_\_\_]_{\sigma}$ X
2. $[\_\_\_]_{\tau}$ Y

are both well-formed predications, then α is a dot object (complex type).

Motivating Dot Objects

When a single word or phrase has the ability to appear in selected contexts that are contradictory in type specification.

If a lexical expression, α, where $\sigma \sqcap \tau = \bot$:

1. $[\_\_\_]_{\sigma}$ X
2. $[\_\_\_]_{\tau}$ Y

are both well-formed predications, then α is a dot object (complex type).

Motivating Dot Objects

When a single word or phrase has the ability to appear in selected contexts that are contradictory in type specification.

If a lexical expression, α, where $\sigma \sqcap \tau = \bot$:

1. $[\_\_\_]_{\sigma}$ X
2. $[\_\_\_]_{\tau}$ Y

are both well-formed predications, then α is a dot object (complex type).

Dot Object Inventory: 1

1. Act•Proposition: promise, allegation, lie
   - I doubt John's promise of marriage.
   - John's promise of marriage happened while we were in Prague.
2. Attribute•Value: temperature, weight, height, tension, strength
   - The temperature is rising.
   - The temperature is 23.

Dot Object Inventory: 1

1. Act•Proposition: promise, allegation, lie
   - I doubt John's promise of marriage.
   - John's promise of marriage happened while we were in Prague.
2. Attribute•Value: temperature, weight, height, tension, strength
   - The temperature is rising.
   - The temperature is 23.

Dot Object Inventory: 1

1. Act•Proposition: promise, allegation, lie
   - I doubt John's promise of marriage.
   - John's promise of marriage happened while we were in Prague.
2. Attribute•Value: temperature, weight, height, tension, strength
   - The temperature is rising.
   - The temperature is 23.

Dot Object Inventory: 2

1. Event•Information: lecture, play, seminar, exam, quiz, test
   a. My lecture lasted an hour.
   b. Nobody understood my lecture.
2. Event•Music: sonata, symphony, song, performance, concert
   a. Mary couldn't hear the concert.
   b. The rain started during the concert.

Dot Object Inventory: 2

1. Event•Information: lecture, play, seminar, exam, quiz, test
   a. My lecture lasted an hour.
   b. Nobody understood my lecture.
2. Event•Music: sonata, symphony, song, performance, concert
   a. Mary couldn't hear the concert.
   b. The rain started during the concert.

Dot Object Inventory: 2

1. Event•Information: lecture, play, seminar, exam, quiz, test
   a. My lecture lasted an hour.
   b. Nobody understood my lecture.
2. Event•Music: sonata, symphony, song, performance, concert
   a. Mary couldn't hear the concert.
   b. The rain started during the concert.

Dot Object Inventory: 3

1. Event•Physical: lunch, breakfast, dinner, tea
   a. My lunch lasted too long today.
   b. I pack my lunch on Thursdays.
2. Information•Physical: book, cd, dvd, dictionary, diary, mail, email, mail, letter
   a. Mary burned my book on Darwin.
   b. Mary believes all of Chomsky's books.

Dot Object Inventory: 3

1. Event•Physical: lunch, breakfast, dinner, tea

   a. My lunch lasted too long today.
   b. I pack my lunch on Thursdays.

2. Information•Physical: book, cd, dvd, dictionary, diary, mail, email, mail, letter

   a. Mary burned my book on Darwin.
   b. Mary believes all of Chomsky's books.

Dot Object Inventory: 3

1. Event•Physical: lunch, breakfast, dinner, tea
   a. My lunch lasted too long today.
   b. I pack my lunch on Thursdays.
2. Information•Physical: book, cd, dvd, dictionary, diary, mail, email, mail, letter
   a. Mary burned my book on Darwin.
   b. Mary believes all of Chomsky's books.

Dot Object Inventory: 4

1. Organization•(Information•Physical): magazine, newspaper, journal
   a. The magazine fired its editor.
   b. The cup is on top of the magazine.
   c. I disagreed with the magazine.
2. Process•Result: construction, depiction, imitation, portrayal, reference
   a. Linnaeus's classification of the species took 25 years.
   b. Linnaeus's classification contains 12,100 species.

Dot Object Inventory: 4

1. Organization•(Information•Physical): magazine, newspaper, journal
   a. The magazine fired its editor.
   b. The cup is on top of the magazine.
   c. I disagreed with the magazine.

2. Process•Result: construction, depiction, imitation, portrayal, reference
   a. Linnaeus's classification of the species took 25 years.
   b. Linnaeus's classification contains 12,100 species.

Dot Object Inventory: 4

1. Organization•(Information•Physical): magazine, newspaper, journal
   a. The magazine fired its editor.
   b. The cup is on top of the magazine.
   c. I disagreed with the magazine.
2. Process•Result: construction, depiction, imitation, portrayal, reference
   a. Linnaeus's classification of the species took 25 years.
   b. Linnaeus's classification contains 12,100 species.

Distinct Principles of Individuation in Dot Objects

1. a. John read every book in the library.
   b. John stole every book in the library.
2. a. Mary answered every question in the class.
   b. Mary repeated every question in the class.

Distinct Principles of Individuation in Dot Objects

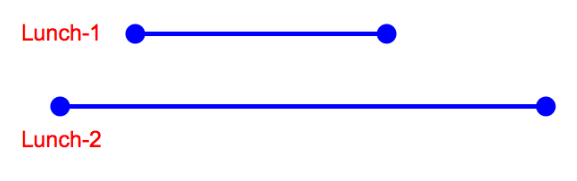
1. a. John read every book in the library.
   b. John stole every book in the library.
2. a. Mary answered every question in the class.
   b. Mary repeated every question in the class.

Distinct Principles of Individuation in Dot Objects

1. a. John read every book in the library.
   b. John stole every book in the library.
2. a. Mary answered every question in the class.
   b. Mary repeated every question in the class.

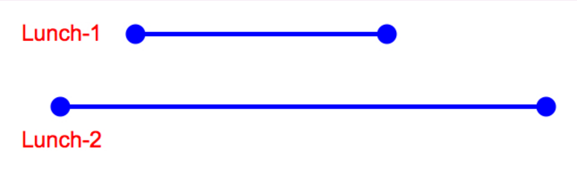
Copredication with Dot Objects: 1

1. Today's lunch$_2$ was longer than yesterday's [__]$_1$.



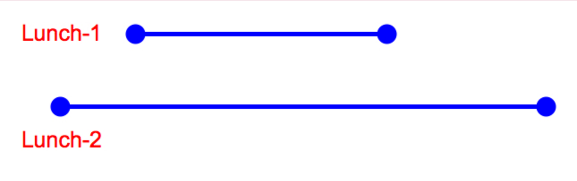
Copredication with Dot Objects: 1

1. Today's lunch$_2$ was longer than yesterday's [__]$_1$.



Copredication with Dot Objects: 1

1. Today's lunch$_2$ was longer than yesterday's [__]$_1$.



Copredication with Dot Objects: 2

1. Today's lunch$_2$ was longer than yesterday's [__]$_1$.

Copredication with Dot Objects: 2

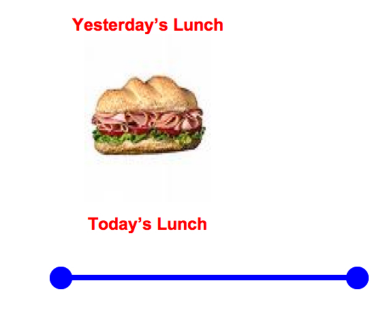
1. Today's lunch$_2$ was longer than yesterday's [___]$_1$.

Copredication with Dot Objects: 2

1. Today's lunch$_2$ was longer than yesterday's [__]$_1$.

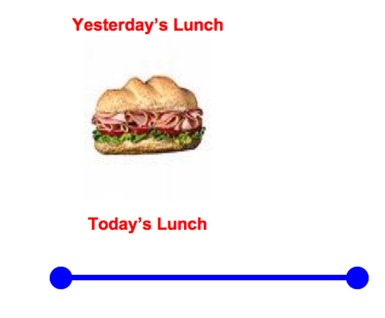
Copredication with Different Dot Object Elements

1. !Today's lunch$_2$ was longer than yesterday's [__]$_1$.



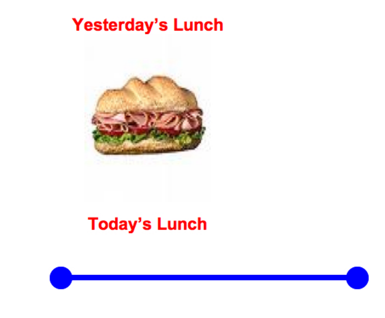
Copredication with Different Dot Object Elements

1. !Today's lunch$_2$ was longer than yesterday's [__]$_1$.



Copredication with Different Dot Object Elements

1. !Today's lunch$_2$ was longer than yesterday's [___]$_1$.



Complex Predicate Types

Predicates formed with a Complex Entity Type as an argument:

1. *read*: $phys \bullet info \rightarrow (e_N \rightarrow t)$
2. Expressed as typed arguments in a λ-expression:
   $\lambda y : phys \bullet info\ \lambda x : e_N[read(x,y)]$
3. Mary read the book.

Complex Predicate Types

Predicates formed with a Complex Entity Type as an argument:

1. $read$: $phys \bullet info \rightarrow (e_N \rightarrow t)$
2. Expressed as typed arguments in a λ-expression:
   $\lambda y : phys \bullet info\ \lambda x : e_N[read(x,y)]$
3. Mary read the book.

Strong Compositionality

If all you have for composition is function application, then you need to create as many lexical entries for an expression as there are environments it appears in. (Weak Compositionality)

Two ways to overcome this:

1. Type Shifting Rules: Geach rule, Rooth and Partee (1982), Partee (1987), Groenendijk and Stokhof (1989).
2. Type Coercion Operations: Moens and Steedman (1988), Pustejovsky (1989), Jacobson (1992), Dölling (1992), Copestake and Briscoe (1992), Hendriks (1993), Egg (1994), Ramsey (1996), de Swart (1998).

Strong Compositionality

If all you have for composition is function application, then you need to create as many lexical entries for an expression as there are environments it appears in. (Weak Compositionality)

Two ways to overcome this:

1. Type Shifting Rules: Geach rule, Rooth and Partee (1982), Partee (1987), Groenendijk and Stokhof (1989).
2. Type Coercion Operations: Moens and Steedman (1988), Pustejovsky (1989), Jacobson (1992), Dölling (1992), Copestake and Briscoe (1992), Hendriks (1993), Egg (1994), Ramsey (1996), de Swart (1998).

Strong Compositionality

If all you have for composition is function application, then you need to create as many lexical entries for an expression as there are environments it appears in. (Weak Compositionality)

Two ways to overcome this:

1. Type Shifting Rules: Geach rule, Rooth and Partee (1982), Partee (1987), Groenendijk and Stokhof (1989).
2. Type Coercion Operations: Moens and Steedman (1988), Pustejovsky (1989), Jacobson (1992), Dölling (1992), Copestake and Briscoe (1992), Hendriks (1993), Egg (1994), Ramsey (1996), de Swart (1998).

Strong Compositionality

If all you have for composition is function application, then you need to create as many lexical entries for an expression as there are environments it appears in. (Weak Compositionality)

Two ways to overcome this:

1. Type Shifting Rules: Geach rule, Rooth and Partee (1982), Partee (1987), Groenendijk and Stokhof (1989).
2. Type Coercion Operations: Moens and Steedman (1988), Pustejovsky (1989), Jacobson (1992), Dölling (1992), Copestake and Briscoe (1992), Hendriks (1993), Egg (1994), Ramsey (1996), de Swart (1998).

Modes of Composition

(9) a. PURE SELECTION (Type Matching): the type a function requires is directly satisfied by the argument;
b. ACCOMMODATION: the type a function requires is inherited by the argument;
c. TYPE COERCION: the type a function requires is imposed on the argument type. This is accomplished by either:
i. *Exploitation*: taking a part of the argument's type to satisfy the function;
ii. *Introduction*: wrapping the argument with the type required by the function.

Modes of Composition

(10) a. PURE SELECTION (Type Matching): the type a function requires is directly satisfied by the argument;
b. ACCOMMODATION: the type a function requires is inherited by the argument;
c. TYPE COERCION: the type a function requires is imposed on the argument type. This is accomplished by either:
i. *Exploitation*: taking a part of the argument's type to satisfy the function;
ii. *Introduction*: wrapping the argument with the type required by the function.

Modes of Composition

(11) a. PURE SELECTION (Type Matching): the type a function requires is directly satisfied by the argument;
b. ACCOMMODATION: the type a function requires is inherited by the argument;
c. TYPE COERCION: the type a function requires is imposed on the argument type. This is accomplished by either:
i. *Exploitation*: taking a part of the argument's type to satisfy the function;
ii. *Introduction*: wrapping the argument with the type required by the function.

Modes of Composition

(12) a. PURE SELECTION (Type Matching): the type a function requires is directly satisfied by the argument;
b. ACCOMMODATION: the type a function requires is inherited by the argument;
c. TYPE COERCION: the type a function requires is imposed on the argument type. This is accomplished by either:
i. *Exploitation*: taking a part of the argument's type to satisfy the function;
ii. *Introduction*: wrapping the argument with the type required by the function.

Modes of Composition

(13) a. PURE SELECTION (Type Matching): the type a function requires is directly satisfied by the argument;
b. ACCOMMODATION: the type a function requires is inherited by the argument;
c. TYPE COERCION: the type a function requires is imposed on the argument type. This is accomplished by either:
i. *Exploitation*: taking a part of the argument's type to satisfy the function;
ii. *Introduction*: wrapping the argument with the type required by the function.

Modes of Composition

(14) a. PURE SELECTION (Type Matching): the type a function requires is directly satisfied by the argument;
b. ACCOMMODATION: the type a function requires is inherited by the argument;
c. TYPE COERCION: the type a function requires is imposed on the argument type. This is accomplished by either:
i. *Exploitation*: taking a part of the argument's type to satisfy the function;
ii. *Introduction*: wrapping the argument with the type required by the function.

Two Kinds of Coercion in Language

- Domain-shifting: The domain of interpretation of the argument is shifted;
- Domain-preserving: The argument is coerced but remains within the general domain of interpretation.

Two Kinds of Coercion in Language

- Domain-shifting: The domain of interpretation of the argument is shifted;
- Domain-preserving: The argument is coerced but remains within the general domain of interpretation.

Two Kinds of Coercion in Language

- Domain-shifting: The domain of interpretation of the argument is shifted;
- Domain-preserving: The argument is coerced but remains within the general domain of interpretation.

Domain-Shifting Coercion

1. Entity shifts to event:
   I enjoyed the beer
2. Entity shifts to proposition:
   I doubt John.

Domain-Shifting Coercion

1. Entity shifts to event:
   I enjoyed the beer
2. Entity shifts to proposition:
   I doubt John.

Domain-Shifting Coercion

1. Entity shifts to event:
   I enjoyed the beer
2. Entity shifts to proposition:
   I doubt John.

Domain-Preserving Coercion

1. Count-mass shifting: There's chicken in the soup.
2. NP Raising: Mary and every child came.
3. Natural-Artifactual shifting: The water spoiled.
4. Natural-Complex shifting: She read a rumor.
5. Complex-Natural shifting: John burnt a book.
6. Artifactual-Natural shifting: She touched the phone.

Domain-Preserving Coercion

1. Count-mass shifting: There's chicken in the soup.
2. NP Raising: Mary and every child came.
3. Natural-Artifactual shifting: The water spoiled.
4. Natural-Complex shifting: She read a rumor.
5. Complex-Natural shifting: John burnt a book.
6. Artifactual-Natural shifting: She touched the phone.

Domain-Preserving Coercion

1. Count-mass shifting: There's chicken in the soup.
2. NP Raising: Mary and every child came.
3. Natural-Artifactual shifting: The water spoiled.
4. Natural-Complex shifting: She read a rumor.
5. Complex-Natural shifting: John burnt a book.
6. Artifactual-Natural shifting: She touched the phone.

Domain-Preserving Coercion

1. Count-mass shifting: There's chicken in the soup.
2. NP Raising: Mary and every child came.
3. Natural-Artifactual shifting: The water spoiled.
4. Natural-Complex shifting: She read a rumor.
5. Complex-Natural shifting: John burnt a book.
6. Artifactual-Natural shifting: She touched the phone.

Domain-Preserving Coercion

1. Count-mass shifting: There's chicken in the soup.
2. NP Raising: Mary and every child came.
3. Natural-Artifactual shifting: The water spoiled.
4. Natural-Complex shifting: She read a rumor.
5. Complex-Natural shifting: John burnt a book.
6. Artifactual-Natural shifting: She touched the phone.

Domain-Preserving Coercion

1. Count-mass shifting: There's chicken in the soup.
2. NP Raising: Mary and every child came.
3. Natural-Artifactual shifting: The water spoiled.
4. Natural-Complex shifting: She read a rumor.
5. Complex-Natural shifting: John burnt a book.
6. Artifactual-Natural shifting: She touched the phone.

Domain-Preserving Coercion

1. Count-mass shifting: There's chicken in the soup.
2. NP Raising: Mary and every child came.
3. Natural-Artifactual shifting: The water spoiled.
4. Natural-Complex shifting: She read a rumor.
5. Complex-Natural shifting: John burnt a book.
6. Artifactual-Natural shifting: She touched the phone.

Direct Argument Selection

- The spokesman denied the statement (PROPOSITION).
- The child threw the ball (PHYSICAL OBJECT).
- The audience didn't believe the rumor (PROPOSITION).

Direct Argument Selection

- The spokesman denied the statement (PROPOSITION).
- The child threw the ball (PHYSICAL OBJECT).
- The audience didn't believe the rumor (PROPOSITION).

Direct Argument Selection

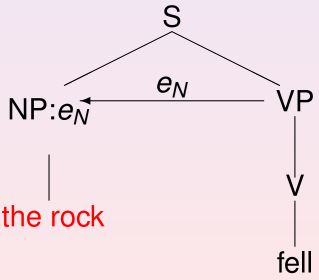
- The spokesman denied the statement (PROPOSITION).
- The child threw the ball (PHYSICAL OBJECT).
- The audience didn't believe the rumor (PROPOSITION).

Direct Argument Selection

- The spokesman denied the statement (PROPOSITION).
- The child threw the ball (PHYSICAL OBJECT).
- The audience didn't believe the rumor (PROPOSITION).

Natural Selection

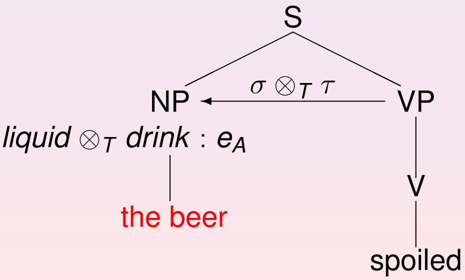
1. The rock fell.



$$\lambda x : e_N[\mathit{fall}(x)]$$

Pure Selection: Artifactual Type

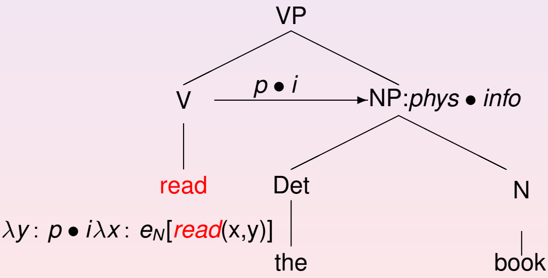
1. The beer spoiled.



$\lambda x : e_A[spoil(x)]$

Pure Selection: Complex Type

1. John read the book.



Coercion of Arguments

- The president denied the attack.
  EVENT $\longrightarrow$ PROPOSITION
- The White House denied this statement.
  LOCATION $\longrightarrow$ HUMAN
- This book explains the theory of relativity.
  PHYS • INFO $\longrightarrow$ human
- d. The Boston office called with an update.
  EVENT $\longrightarrow$ INFO

Coercion of Arguments

- The president denied the attack.
  EVENT → PROPOSITION
- The White House denied this statement.
  LOCATION → HUMAN
- This book explains the theory of relativity.
  PHYS • INFO → human
- d. The Boston office called with an update.
  EVENT → INFO

Coercion of Arguments

- The president denied the attack.
  EVENT $\longrightarrow$ PROPOSITION
- The White House denied this statement.
  LOCATION $\longrightarrow$ HUMAN
- This book explains the theory of relativity.
  PHYS • INFO $\longrightarrow$ human
- d. The Boston office called with an update.
  EVENT $\longrightarrow$ INFO

Coercion of Arguments

- The president denied the attack.
  EVENT → PROPOSITION
- The White House denied this statement.
  LOCATION → HUMAN
- This book explains the theory of relativity.
  PHYS • INFO → human
- d. The Boston office called with an update.
  EVENT → INFO

Coercion of Arguments

- The president denied the attack.
  EVENT $\longrightarrow$ PROPOSITION
- The White House denied this statement.
  LOCATION $\longrightarrow$ HUMAN
- This book explains the theory of relativity.
  PHYS • INFO $\longrightarrow$ human
- d. The Boston office called with an update.
  EVENT $\longrightarrow$ INFO

Coercion of Arguments

- The president denied the attack.
  EVENT → PROPOSITION
- The White House denied this statement.
  LOCATION → HUMAN
- This book explains the theory of relativity.
  PHYS • INFO → human
- d. The Boston office called with an update.
  EVENT → INFO

Coercion of Arguments

- The president denied the attack.
  EVENT $\longrightarrow$ PROPOSITION
- The White House denied this statement.
  LOCATION $\longrightarrow$ HUMAN
- This book explains the theory of relativity.
  PHYS • INFO $\longrightarrow$ human
- d. The Boston office called with an update.
  EVENT $\longrightarrow$ INFO

Coercion of Arguments

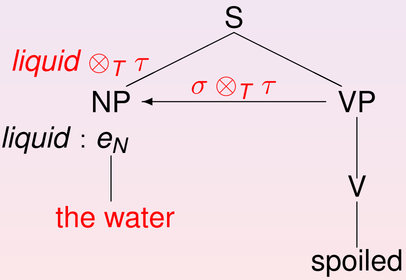
- The president denied the attack.
  EVENT $\longrightarrow$ PROPOSITION
- The White House denied this statement.
  LOCATION $\longrightarrow$ HUMAN
- This book explains the theory of relativity.
  PHYS • INFO $\longrightarrow$ human
- d. The Boston office called with an update.
  EVENT $\longrightarrow$ INFO

Coercion of Arguments

- The president denied the attack.
  EVENT $\longrightarrow$ PROPOSITION
- The White House denied this statement.
  LOCATION $\longrightarrow$ HUMAN
- This book explains the theory of relativity.
  PHYS $\bullet$ INFO $\longrightarrow$ human
- d. The Boston office called with an update.
  EVENT $\longrightarrow$ INFO

Type Coercion: Qualia-Introduction

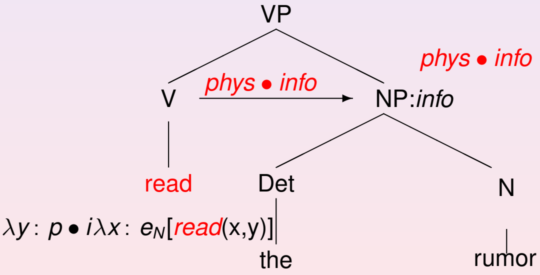
1. The water spoiled.



$\lambda x : e_A[spoil(x)]$

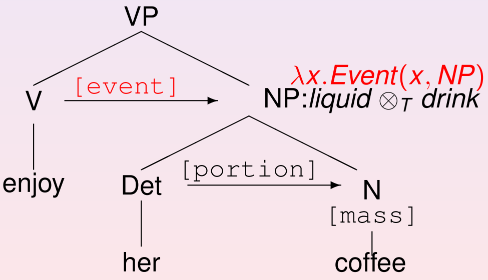
Type Coercion: Natural to Complex Introduction

John read the rumor.



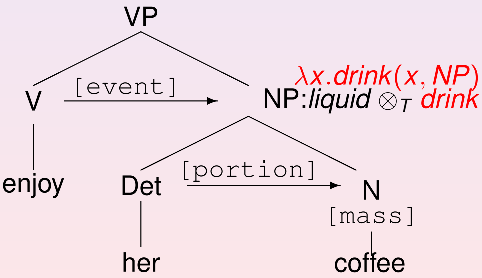
Type Coercion: Event Introduction

1. Mary enjoyed her coffee.



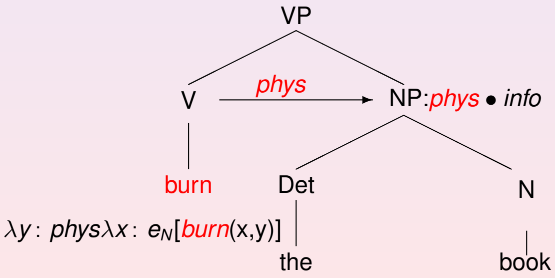
Type Coercion: Qualia Exploitation

1. Mary enjoyed her coffee.



Type Coercion: Dot Exploitation

1. The police burned the book.
2. Mary believes the book.



Verb-Argument Composition Table

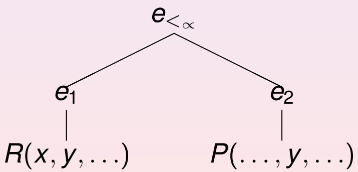
	Verb selects:		
Argument is:	**Natural**	**Artifactual**	**Complex**
Natural	Selection	Qualia Intro	Dot Intro
Artifactual	Qualia Exploit	Selection	Dot Intro
Complex	Dot Exploit	Dot Exploit	Selextion

Interpreting the Subject in Causatives

- Assume a causative (binary) event structure
- Argument selection:
  - subject is event:
    $e \rightarrow (\epsilon \rightarrow t)$
  - subject is entity:
    $e \rightarrow (e \rightarrow t)$

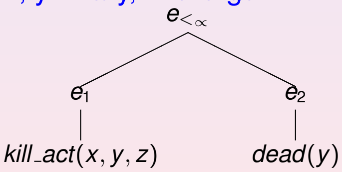
Causative Argument Coherence

1. The relation identified as the initial event and that identified as the resulting event must refer to at least one argument in common.



Coercion of the External Argument

1. If the DP is a direct argument to event, e_1, then an interpretation is possible through a coercion.
2. $kill\_act(e_1, x, y, z)$
3. x=John, y=Mary, z=the-gun



Satisfaction of event typing is achieved by exploiting the argument and wrapping it with the event it participates in.

Introducing Agency over Predicates

Wechsler's Subject Rule is a factor of inherent agency of the argument.

1. John rolled down the hill as fast as he could.
2. John cooled off with an iced latte.

- Human is typed as an acting, rational, animal: *human* $\otimes_A \sigma \otimes_T \tau$

Introducing Agency over Predicates

Wechsler's Subject Rule is a factor of inherent agency of the argument.

1. John rolled down the hill as fast as he could.
2. John cooled off with an iced latte.

- Human is typed as an acting, rational, animal: $human \otimes_A \sigma \otimes_T \tau$

Introducing Agency over Predicates

Wechsler's Subject Rule is a factor of inherent agency of the argument.

1. John rolled down the hill as fast as he could.
2. John cooled off with an iced latte.

- Human is typed as an acting, rational, animal: *human* $\otimes_A \sigma \otimes_T \tau$

Introducing Agency over Predicates

Wechsler's Subject Rule is a factor of inherent agency of the argument.

1. John rolled down the hill as fast as he could.
2. John cooled off with an iced latte.

- Human is typed as an acting, rational, animal: $human \otimes_A \sigma \otimes_T \tau$

Perception Predicates

The verb hear selects for the type SOUND.

- $sound \rightarrow (anim \rightarrow t)$
- Conventionalized Attributes of an object:

1. $sound(dog)$ = barking, whining
2. $sound(rain)$ = falling, hitting the roof

Perception Predicates

The verb hear selects for the type SOUND.

- $sound \rightarrow (anim \rightarrow t)$
- Conventionalized Attributes of an object:

1. $sound(dog)$ = barking, whining
2. $sound(rain)$ = falling, hitting the roof

Perception Predicates

The verb hear selects for the type SOUND.

- *sound* $\rightarrow$ (*anim* $\rightarrow$ *t*)
- Conventionalized Attributes of an object:

1. *sound*(*dog*) = barking, whining
2. *sound*(*rain*) = falling, hitting the roof

Perception Predicates

The verb hear selects for the type SOUND.

- *sound* → (*anim* → *t*)
- Conventionalized Attributes of an object:

1. *sound*(*dog*) = barking, whining
2. *sound*(*rain*) = falling, hitting the roof

Perception Predicates

The verb hear selects for the type SOUND.

- $sound \rightarrow (anim \rightarrow t)$
- Conventionalized Attributes of an object:

1. $sound(dog)$ = barking, whining
2. $sound(rain)$ = falling, hitting the roof

Encoding Change through Selection

1. a. Mary fixed every leaky faucet.
   b. Mary fixed every brass faucet.
2. a. John drank a full glass of milk.
   b. !John drank an empty glass of milk.
3. John closed the open door.
4. People filled the empty hall.
5. a. Mary cleaned the dirty table.
   b. Mary cleaned the glass table.
6. a. [The audience]$_i$ left the theatre.
   b. *[It]$_i$ went home.
   c. [They]$_i$ went home.

Encoding Change through Selection

1. a. Mary fixed every leaky faucet.
   b. Mary fixed every brass faucet.
2. a. John drank a full glass of milk.
   b. !John drank an empty glass of milk.
3. John closed the open door.
4. People filled the empty hall.
5. a. Mary cleaned the dirty table.
   b. Mary cleaned the glass table.
6. a. [The audience]$_i$ left the theatre.
   b. *[It]$_i$ went home.
   c. [They]$_i$ went home.

Encoding Change through Selection

1. a. Mary fixed every leaky faucet.
   b. Mary fixed every brass faucet.
2. a. John drank a full glass of milk.
   b. !John drank an empty glass of milk.
3. John closed the open door.
4. People filled the empty hall.
5. a. Mary cleaned the dirty table.
   b. Mary cleaned the glass table.
6. a. [The audience]$_i$ left the theatre.
   b. *[It]$_i$ went home.
   c. [They]$_i$ went home.

Encoding Change through Selection

1. a. Mary fixed every leaky faucet.
   b. Mary fixed every brass faucet.
2. a. John drank a full glass of milk.
   b. !John drank an empty glass of milk.
3. John closed the open door.
4. People filled the empty hall.
5. a. Mary cleaned the dirty table.
   b. Mary cleaned the glass table.
6. a. [The audience]$_i$ left the theatre.
   b. *[It]$_i$ went home.
   c. [They]$_i$ went home.

Encoding Change through Selection

1. a. Mary fixed every leaky faucet.
   b. Mary fixed every brass faucet.
2. a. John drank a full glass of milk.
   b. !John drank an empty glass of milk.
3. John closed the open door.
4. People filled the empty hall.
5. a. Mary cleaned the dirty table.
   b. Mary cleaned the glass table.
6. a. [The audience]$_i$ left the theatre.
   b. *[It]$_i$ went home.
   c. [They]$_i$ went home.

Encoding Change through Selection

1. a. Mary fixed every leaky faucet.
   b. Mary fixed every brass faucet.
2. a. John drank a full glass of milk.
   b. !John drank an empty glass of milk.
3. John closed the open door.
4. People filled the empty hall.
5. a. Mary cleaned the dirty table.
   b. Mary cleaned the glass table.
6. a. [The audience]$_i$ left the theatre.
   b. *[It]$_i$ went home.
   c. [They]$_i$ went home.

Encoding Change through Selection

1. a. Mary fixed every leaky faucet.
   b. Mary fixed every brass faucet.
2. a. John drank a full glass of milk.
   b. !John drank an empty glass of milk.
3. John closed the open door.
4. People filled the empty hall.
5. a. Mary cleaned the dirty table.
   b. Mary cleaned the glass table.
6. a. [The audience]$_i$ left the theatre.
   b. *[It]$_i$ went home.
   c. [They]$_i$ went home.

Modeling Change

1. Situations: s, how the world may be described;
2. Fluents: f, time-varying properties of individuals;
3. Actions: a, operators that change the value of fluents.
4. cf. van Lambalgen and Hamm (2005)

- Effect Axioms: take into account the preconditions of an action for it to happen;
- Frame Axioms: take into account what does not change with an action
- Order of $f \times a$ frame axioms for a given domain.

Modeling Change

1. Situations: *s*, how the world may be described;
2. Fluents: *f*, time-varying properties of individuals;
3. Actions: *a*, operators that change the value of fluents.
4. cf. van Lambalgen and Hamm (2005)

- Effect Axioms: take into account the preconditions of an action for it to happen;
- Frame Axioms: take into account what does not change with an action
- Order of $f \times a$ frame axioms for a given domain.

Modeling Change

1. Situations: *s*, how the world may be described;
2. Fluents: *f*, time-varying properties of individuals;
3. Actions: *a*, operators that change the value of fluents.
4. cf. van Lambalgen and Hamm (2005)

- Effect Axioms: take into account the preconditions of an action for it to happen;
- Frame Axioms: take into account what does not change with an action
- Order of $f \times a$ frame axioms for a given domain.

Modeling Change

1. Situations: *s*, how the world may be described;
2. Fluents: *f*, time-varying properties of individuals;
3. Actions: *a*, operators that change the value of fluents.
4. cf. van Lambalgen and Hamm (2005)

- Effect Axioms: take into account the preconditions of an action for it to happen;
- Frame Axioms: take into account what does not change with an action
- Order of $f \times a$ frame axioms for a given domain.

Modeling Change

1. Situations: s, how the world may be described;
2. Fluents: f, time-varying properties of individuals;
3. Actions: a, operators that change the value of fluents.
4. cf. van Lambalgen and Hamm (2005)

- Effect Axioms: take into account the preconditions of an action for it to happen;
- Frame Axioms: take into account what does not change with an action
- Order of $f \times a$ frame axioms for a given domain.

Modeling Change

1. Situations: *s*, how the world may be described;
2. Fluents: *f*, time-varying properties of individuals;
3. Actions: *a*, operators that change the value of fluents.
4. cf. van Lambalgen and Hamm (2005)

- Effect Axioms: take into account the preconditions of an action for it to happen;
- Frame Axioms: take into account what does not change with an action
- Order of $f \times a$ frame axioms for a given domain.

Modeling Change

1. Situations: s, how the world may be described;
2. Fluents: f, time-varying properties of individuals;
3. Actions: a, operators that change the value of fluents.
4. cf. van Lambalgen and Hamm (2005)

- Effect Axioms: take into account the preconditions of an action for it to happen;
- Frame Axioms: take into account what does not change with an action
- Order of $f \times a$ frame axioms for a given domain.

Stateless and State-based Selection

1. Stateless Selection: Selection is performed independent of the operation performed on the argument. Selection is without state information.
2. Event-based Selection: Selection is performed over a trace of the operation performed on the argument. Selection is sensitive to the states the argument participates in.

Stateless and State-based Selection

1. Stateless Selection: Selection is performed independent of the operation performed on the argument. Selection is without state information.
2. Event-based Selection: Selection is performed over a trace of the operation performed on the argument. Selection is sensitive to the states the argument participates in.

Dynamic Typing (cf. Steedman, 2000)

1. Whenever the predicate α is interpreted succesfully, ϕ holds in the discourse.
   $[\alpha]\phi$
2. It is possible to interpret the predicate α such that ϕ holds in the discourse.
   $\langle\alpha\rangle\phi$
3. Given the precondition of ψ, whenever the predicate α is interpreted succesfully, ϕ holds in the discourse.
   $\psi \rightarrow [\alpha]\phi$

Dynamic Typing (cf. Steedman, 2000)

1. Whenever the predicate α is interpreted succesfully, ϕ holds in the discourse.
   $[\alpha]\phi$
2. It is possible to interpret the predicate α such that ϕ holds in the discourse.
   $\langle\alpha\rangle\phi$
3. Given the precondition of ψ, whenever the predicate α is interpreted succesfully, ϕ holds in the discourse.
   $\psi \rightarrow [\alpha]\phi$

Dynamic Typing (cf. Steedman, 2000)

1. Whenever the predicate α is interpreted succesfully, ϕ holds in the discourse.
   $[\alpha]\phi$
2. It is possible to interpret the predicate α such that ϕ holds in the discourse.
   $\langle\alpha\rangle\phi$
3. Given the precondition of ψ, whenever the predicate α is interpreted succesfully, ϕ holds in the discourse.
   $\psi \rightarrow [\alpha]\phi$

Lexicalizing the statement of change

1. kill: $\neg dead(y) \rightarrow [kill(x, y)]dead(y)$
2. break: $\neg broken(y) \rightarrow [break(x, y)]broken(y)$
3. fill: $[fill(x, y)]full(y)$

Stateless Selection with Dynamic Interpretation:

a. *kill*: $anim \rightarrow (e_N \rightarrow t)$

b. $\lambda y : anim\ \lambda x : e_N(\neg dead(y)[kill(x, y]dead(y))$

Lexicalizing the statement of change

1. kill: $\neg dead(y) \rightarrow [kill(x,y)]dead(y)$
2. break: $\neg broken(y) \rightarrow [break(x,y)]broken(y)$
3. fill: $[fill(x,y)]full(y)$

Stateless Selection with Dynamic Interpretation:

a. $kill$: $anim \rightarrow (e_N \rightarrow t)$

b. $\lambda y : anim\ \lambda x : e_N(\neg dead(y)[kill(x,y]dead(y))$

Lexicalizing the statement of change

1. kill: $\neg dead(y) \rightarrow [kill(x, y)]dead(y)$
2. break: $\neg broken(y) \rightarrow [break(x, y)]broken(y)$
3. fill: $[fill(x, y)]full(y)$

Stateless Selection with Dynamic Interpretation:

a. $kill$: $anim \rightarrow (e_N \rightarrow t)$

b. $\lambda y : anim\ \lambda x : e_N(\neg dead(y)[kill(x, y]dead(y))$

Lexicalizing the statement of change

1. kill: $\neg dead(y) \rightarrow [kill(x, y)]dead(y)$
2. break: $\neg broken(y) \rightarrow [break(x, y)]broken(y)$
3. fill: $[fill(x, y)]full(y)$

Stateless Selection with Dynamic Interpretation:

a. $kill$: $anim \rightarrow (e_N \rightarrow t)$

b. $\lambda y : anim\ \lambda x : e_N(\neg dead(y)[kill(x, y]dead(y))$

Lexicalizing the statement of change

1. kill: $\neg dead(y) \rightarrow [kill(x,y)]dead(y)$
2. break: $\neg broken(y) \rightarrow [break(x,y)]broken(y)$
3. fill: $[fill(x,y)]full(y)$

Stateless Selection with Dynamic Interpretation:

a. $kill$: $anim \rightarrow (e_N \rightarrow t)$

b. $\lambda y : anim\ \lambda x : e_N(\neg dead(y)[kill(x,y]dead(y))$

Lexicalizing the statement of change

1. kill: $\neg dead(y) \rightarrow [kill(x, y)]dead(y)$
2. break: $\neg broken(y) \rightarrow [break(x, y)]broken(y)$
3. fill: $[fill(x, y)]full(y)$

Stateless Selection with Dynamic Interpretation:

a. $kill$: $anim \rightarrow (e_N \rightarrow t)$

b. $\lambda y : anim\ \lambda x : e_N(\neg dead(y)[kill(x, y]dead(y))$

State-based Selection (Pustejovsky, 2007)

Let $\bar{\alpha}$ refer to the trace of the type α through the event structure, $\mathcal{E}$, associated with the predicate, $\bar{\alpha} \rightarrow t$. The trace is an array of indices associated with the type. A predicate can select either type $\bar{\alpha}$ or α (cf. Löbner, 1981, Romero, 2008)

1. Stateless: $\alpha \rightarrow t$. Reference only to the argument.
2. State-based: $\bar{\alpha} \rightarrow t$. Reference to the trace of the argument through the event structure.

a. The temperature is 25 degrees: $e \rightarrow t$

b. The temperature is rising: $\bar{e} \rightarrow t$

State-based Selection (Pustejovsky, 2007)

Let $\bar{\alpha}$ refer to the trace of the type α through the event structure, $\mathcal{E}$, associated with the predicate, $\bar{\alpha} \rightarrow t$. The trace is an array of indices associated with the type. A predicate can select either type $\bar{\alpha}$ or α (cf. Löbner, 1981, Romero, 2008)

1. Stateless: $\alpha \rightarrow t$. Reference only to the argument.
2. State-based: $\bar{\alpha} \rightarrow t$. Reference to the trace of the argument through the event structure.

a. The temperature is 25 degrees: $e \rightarrow t$

b. The temperature is rising: $\bar{e} \rightarrow t$

State-based Selection (Pustejovsky, 2007)

Let $\bar{\alpha}$ refer to the trace of the type α through the event structure, $\mathcal{E}$, associated with the predicate, $\bar{\alpha} \rightarrow t$. The trace is an array of indices associated with the type. A predicate can select either type $\bar{\alpha}$ or α (cf. Löbner, 1981, Romero, 2008)

1. Stateless: $\alpha \rightarrow t$. Reference only to the argument.
2. State-based: $\bar{\alpha} \rightarrow t$. Reference to the trace of the argument through the event structure.

a. The temperature is 25 degrees: $e \rightarrow t$

b. The temperature is rising: $\bar{e} \rightarrow t$

State-based Selection (Pustejovsky, 2007)

Let $\bar{\alpha}$ refer to the trace of the type α through the event structure, $\mathcal{E}$, associated with the predicate, $\bar{\alpha} \rightarrow t$. The trace is an array of indices associated with the type. A predicate can select either type $\bar{\alpha}$ or α (cf. Löbner, 1981, Romero, 2008)

1. Stateless: $\alpha \rightarrow t$. Reference only to the argument.
2. State-based: $\bar{\alpha} \rightarrow t$. Reference to the trace of the argument through the event structure.

a. The temperature is 25 degrees: $e \rightarrow t$

b. The temperature is rising: $\bar{e} \rightarrow t$

State-based Selection (Pustejovsky, 2007)

Let $\bar{\alpha}$ refer to the trace of the type α through the event structure, $\mathcal{E}$, associated with the predicate, $\bar{\alpha} \rightarrow t$. The trace is an array of indices associated with the type. A predicate can select either type $\bar{\alpha}$ or α (cf. Löbner, 1981, Romero, 2008)

1. Stateless: $\alpha \rightarrow t$. Reference only to the argument.
2. State-based: $\bar{\alpha} \rightarrow t$. Reference to the trace of the argument through the event structure.

a. The temperature is 25 degrees: $e \rightarrow t$

b. The temperature is rising: $\bar{e} \rightarrow t$

Gating Predicate

Traces let us refer to change as an aspect of the type of the predicate. Hence, a change predicate has a different functional type from a stateless predicate.

1. Gates:
   Let us define a pair of type operators, $\ulcorner$ and $\urcorner$, applied over a trace, that initiate or terminate a process or state. We will call the resulting transformations, *gating functions*.

Gating Predicate

Traces let us refer to change as an aspect of the type of the predicate. Hence, a change predicate has a different functional type from a stateless predicate.

1. Gates:
   Let us define a pair of type operators, $\ulcorner$ and $\urcorner$, applied over a trace, that initiate or terminate a process or state. We will call the resulting transformations, *gating functions*.

State-based Typing: 1

1. Refer to the trace of the argument:
   If $a \rightarrow b$ is a type, then $\bar{a} \rightarrow b$ is a type.
2. Initiate or terminate the argument:
   If $\bar{a} \rightarrow b$ is a type, then $\ulcorner a \rightarrow b$ and $a\urcorner \rightarrow b$ are types.
3. Initiate or terminate a qualia value:
   If $a \otimes c \rightarrow b$ is a type, then $a \otimes \ulcorner c \rightarrow b$ and $a \otimes c\urcorner \rightarrow b$ are types.
4. Initiate or terminate a dot element:
   If $a \bullet c \rightarrow b$ is a type, then $\ulcorner a \bullet c \rightarrow b$ and $a\urcorner \bullet c \rightarrow b$ are types.

State-based Typing: 1

1. Refer to the trace of the argument:
   If $a \rightarrow b$ is a type, then $\bar{a} \rightarrow b$ is a type.
2. Initiate or terminate the argument:
   If $\bar{a} \rightarrow b$ is a type, then $\ulcorner a \rightarrow b$ and $a\urcorner \rightarrow b$ are types.
3. Initiate or terminate a qualia value:
   If $a \otimes c \rightarrow b$ is a type, then $a \otimes \ulcorner c \rightarrow b$ and $a \otimes c\urcorner \rightarrow b$ are types.
4. Initiate or terminate a dot element:
   If $a \bullet c \rightarrow b$ is a type, then $\ulcorner a \bullet c \rightarrow b$ and $a\urcorner \bullet c \rightarrow b$ are types.

State-based Typing: 1

1. Refer to the trace of the argument:
   If $a \rightarrow b$ is a type, then $\bar{a} \rightarrow b$ is a type.
2. Initiate or terminate the argument:
   If $\bar{a} \rightarrow b$ is a type, then $\ulcorner a \rightarrow b$ and $a^{\urcorner} \rightarrow b$ are types.
3. Initiate or terminate a qualia value:
   If $a \otimes c \rightarrow b$ is a type, then $a \otimes \ulcorner c \rightarrow b$ and $a \otimes c^{\urcorner} \rightarrow b$ are types.
4. Initiate or terminate a dot element:
   If $a \bullet c \rightarrow b$ is a type, then $\ulcorner a \bullet c \rightarrow b$ and $a^{\urcorner} \bullet c \rightarrow b$ are types.

State-based Typing: 1

1. Refer to the trace of the argument:
   If $a \rightarrow b$ is a type, then $\bar{a} \rightarrow b$ is a type.
2. Initiate or terminate the argument:
   If $\bar{a} \rightarrow b$ is a type, then $\ulcorner a \rightarrow b$ and $a\urcorner \rightarrow b$ are types.
3. Initiate or terminate a qualia value:
   If $a \otimes c \rightarrow b$ is a type, then $a \otimes \ulcorner c \rightarrow b$ and $a \otimes c\urcorner \rightarrow b$ are types.
4. Initiate or terminate a dot element:
   If $a \bullet c \rightarrow b$ is a type, then $\ulcorner a \bullet c \rightarrow b$ and $a\urcorner \bullet c \rightarrow b$ are types.

State-based Typing: 1

1. Refer to the trace of the argument:
   If $a \rightarrow b$ is a type, then $\bar{a} \rightarrow b$ is a type.
2. Initiate or terminate the argument:
   If $\bar{a} \rightarrow b$ is a type, then $\ulcorner a \rightarrow b$ and $a \urcorner \rightarrow b$ are types.
3. Initiate or terminate a qualia value:
   If $a \otimes c \rightarrow b$ is a type, then $a \otimes \ulcorner c \rightarrow b$ and $a \otimes c \urcorner \rightarrow b$ are types.
4. Initiate or terminate a dot element:
   If $a \bullet c \rightarrow b$ is a type, then $\ulcorner a \bullet c \rightarrow b$ and $a \urcorner \bullet c \rightarrow b$ are types.

State-based Typing: 2

1. a. The door opened.
   b. The window closed.
2. Predicates as State-based Transition Functions:
   a. *open*: $phys \bullet \ulcorner aperture \rightarrow t$
   b. *close*: $phys \bullet aperture \urcorner \rightarrow t$

State-based Typing: 2

1. a. The door opened.
   b. The window closed.
2. Predicates as State-based Transition Functions:
   a. *open*: $phys \bullet \ulcorner aperture \rightarrow t$
   b. *close*: $phys \bullet aperture \urcorner \rightarrow t$

State-based Typing: 2

1. a. The door opened.
   b. The window closed.
2. Predicates as State-based Transition Functions:
   a. *open*: $phys \bullet \ulcorner aperture \rightarrow t$
   b. *close*: $phys \bullet aperture \urcorner \rightarrow t$

State-based Typing: 3

1. Gates over Natural
   - animal: be born, die
   - apple: grow, rot
2. Gates over Artifactual
   - prisoner: arrest, escape
   - audience: assemble, disperse
   - cake: bake, eat
3. Gates over Complex
   i. door: *phys* • *aperture*:
      build(phys), destroy(phys),
      open(aperture), close(aperture)
   ii. talk: *event* • *info*:
      begin(event), end(event),
      prepare(info)

State-based Typing: 3

1. Gates over Natural
   animal: be born, die
   apple: grow, rot
2. Gates over Artifactual
   prisoner: arrest, escape
   audience: assemble, disperse
   cake: bake, eat
3. Gates over Complex
   i. door: *phys* • *aperture*:
   build(phys), destroy(phys),
   open(aperture), close(aperture)
   ii. talk: *event* • *info*:
   begin(event), end(event),
   prepare(info)

State-based Typing: 3

1. Gates over Natural
   - animal: be born, die
   - apple: grow, rot
2. Gates over Artifactual
   - prisoner: arrest, escape
   - audience: assemble, disperse
   - cake: bake, eat
3. Gates over Complex
   - i. door: *phys* • *aperture*: build(phys), destroy(phys), open(aperture), close(aperture)
   - ii. talk: *event* • *info*: begin(event), end(event), prepare(info)

State-based Typing: 3

1. Gates over Natural
   animal: be born, die
   apple: grow, rot
2. Gates over Artifactual
   prisoner: arrest, escape
   audience: assemble, disperse
   cake: bake, eat
3. Gates over Complex
   i. door: *phys* • *aperture*:
   build(phys), destroy(phys),
   open(aperture), close(aperture)
   ii. talk: *event* • *info*:
   begin(event), end(event),
   prepare(info)

Derivation involving state-based selection

1. an escaped prisoner
2. $\lambda P \lambda x \exists e[\text{escaped}(e, x) \wedge P(e, x)]$
   x: p;
   escaped: $(p \otimes captive \rightarrow t) \rightarrow (p \otimes captive^{\neg} \rightarrow t)$.
3. $\lambda v \text{prisoner}(v)$: $(human \otimes captive) \rightarrow t$
4. $[\![\text{escaped}]\!] =$
   $\lambda P \lambda x \lambda e_2 \exists e_1[\neg captive(e_2, x) \wedge captive(e_1, x) \wedge e_1 < e_2 \wedge P(e_2, x)]$;
5. $[\![\text{prisoner}]\!] = \lambda x \lambda e[human(x,) \wedge captive(e, x)]$
6. $[\![\text{escaped prisoner}]\!] =$
   $\lambda x \lambda e_2 \exists e_1[\neg captive(e_2, x) \wedge captive(e_1, x) \wedge e_1 < e_2 \wedge human(e_2, x)]$

Derivation involving state-based selection

1. an escaped prisoner
2. $\lambda P \lambda x \exists e[\text{escaped}(e, x) \wedge P(e, x)]$
   x: p;
   escaped: $(p \otimes captive \rightarrow t) \rightarrow (p \otimes captive^{\neg} \rightarrow t)$.
3. $\lambda v \text{prisoner}(v)$: $(human \otimes captive) \rightarrow t$
4. $[\![\text{escaped}]\!] =$
   $\lambda P \lambda x \lambda e_2 \exists e_1 [\neg captive(e_2, x) \wedge captive(e_1, x) \wedge e_1 < e_2 \wedge P(e_2, x)]$;
5. $[\![\text{prisoner}]\!] = \lambda x \lambda e[human(x,) \wedge captive(e, x)]$
6. $[\![\text{escaped prisoner}]\!] =$
   $\lambda x \lambda e_2 \exists e_1 [\neg captive(e_2, x) \wedge captive(e_1, x) \wedge e_1 < e_2 \wedge human(e_2, x)]$

Derivation involving state-based selection

1. an escaped prisoner
2. $\lambda P \lambda x \exists e[\text{escaped}(e, x) \wedge P(e, x)]$
   x: p;
   escaped: $(p \otimes captive \rightarrow t) \rightarrow (p \otimes captive^{\neg} \rightarrow t)$.
3. $\lambda v \text{prisoner}(v)$: $(human \otimes captive) \rightarrow t$
4. $[\![\text{escaped}]\!] =$
   $\lambda P \lambda x \lambda e_2 \exists e_1[\neg captive(e_2, x) \ \wedge \ \ captive(e_1, x) \wedge e_1 <$
   $e_2 \wedge P(e_2, x)]$;
5. $[\![\text{prisoner}]\!] = \lambda x \lambda e[human(x,) \wedge captive(e, x)]$
6. $[\![\text{escaped prisoner}]\!] =$
   $\lambda x \lambda e_2 \exists e_1[\neg captive(e_2, x) \ \wedge \ \ captive(e_1, x) \wedge e_1 <$
   $e_2 \wedge human(e_2, x)]$

Derivation involving state-based selection

1. an escaped prisoner
2. $\lambda P \lambda x \exists e[\text{escaped}(e, x) \wedge P(e, x)]$
   x: p;
   escaped: $(p \otimes captive \rightarrow t) \rightarrow (p \otimes captive^{\neg} \rightarrow t)$.
3. $\lambda v \text{prisoner}(v)$: $(human \otimes captive) \rightarrow t$
4. $[\![\text{escaped}]\!] =$
   $\lambda P \lambda x \lambda e_2 \exists e_1[\neg captive(e_2, x) \wedge captive(e_1, x) \wedge e_1 < e_2 \wedge P(e_2, x)]$;
5. $[\![\text{prisoner}]\!] = \lambda x \lambda e[human(x,) \wedge captive(e, x)]$
6. $[\![\text{escaped prisoner}]\!] =$
   $\lambda x \lambda e_2 \exists e_1[\neg captive(e_2, x) \wedge captive(e_1, x) \wedge e_1 < e_2 \wedge human(e_2, x)]$

Derivation involving state-based selection

1. an escaped prisoner
2. $\lambda P \lambda x \exists e[\text{escaped}(e, x) \wedge P(e, x)]$
   x: p;
   escaped: $(p \otimes captive \rightarrow t) \rightarrow (p \otimes captive^{\neg} \rightarrow t)$.
3. $\lambda v \text{prisoner}(v)$: $(human \otimes captive) \rightarrow t$
4. $[\![\text{escaped}]\!] =$
   $\lambda P \lambda x \lambda e_2 \exists e_1[\neg captive(e_2, x) \wedge captive(e_1, x) \wedge e_1 < e_2 \wedge P(e_2, x)]$;
5. $[\![\text{prisoner}]\!] = \lambda x \lambda e[human(x,) \wedge captive(e, x)]$
6. $[\![\text{escaped prisoner}]\!] =$
   $\lambda x \lambda e_2 \exists e_1[\neg captive(e_2, x) \wedge captive(e_1, x) \wedge e_1 < e_2 \wedge human(e_2, x)]$

Derivation involving state-based selection

1. an escaped prisoner
2. $\lambda P \lambda x \exists e[\text{escaped}(e, x) \wedge P(e, x)]$
   x: p;
   escaped: $(p \otimes captive \rightarrow t) \rightarrow (p \otimes captive^{\neg} \rightarrow t)$.
3. $\lambda v \text{prisoner}(v)$: $(human \otimes captive) \rightarrow t$
4. $[\![\text{escaped}]\!] =$
   $\lambda P \lambda x \lambda e_2 \exists e_1[\neg captive(e_2, x) \ \wedge \ \ captive(e_1, x) \wedge e_1 < e_2 \wedge P(e_2, x)]$;
5. $[\![\text{prisoner}]\!] = \lambda x \lambda e[human(x,) \wedge captive(e, x)]$
6. $[\![\text{escaped prisoner}]\!] =$
   $\lambda x \lambda e_2 \exists e_1[\neg captive(e_2, x) \ \wedge \ \ captive(e_1, x) \wedge e_1 < e_2 \wedge human(e_2, x)]$

Conclusion

1. There are two kinds of polysemy:
   1. inherent polysemy
   2. selectional polysemy
2. Mechanisms of Selection in language involve:
   1. function application
   2. type coercion by exploitation
   3. type coercion by introduction
   4. type accommodation
3. Predicates can be encoded as selecting an argument typed
   1. statelessly
   2. state-based

Conclusion

1. There are two kinds of polysemy:
   1. inherent polysemy
   2. selectional polysemy
2. Mechanisms of Selection in language involve:
   1. function application
   2. type coercion by exploitation
   3. type coercion by introduction
   4. type accommodation
3. Predicates can be encoded as selecting an argument typed
   1. statelessly
   2. state-based

Conclusion

1. There are two kinds of polysemy:
   1. inherent polysemy
   2. selectional polysemy
2. Mechanisms of Selection in language involve:
   1. function application
   2. type coercion by exploitation
   3. type coercion by introduction
   4. type accommodation
3. Predicates can be encoded as selecting an argument typed
   1. statelessly
   2. state-based

Conclusion

1. There are two kinds of polysemy:
   1. inherent polysemy
   2. selectional polysemy
2. Mechanisms of Selection in language involve:
   1. function application
   2. type coercion by exploitation
   3. type coercion by introduction
   4. type accommodation
3. Predicates can be encoded as selecting an argument typed
   1. statelessly
   2. state-based