
CSci 5541: Natural Language Processing

Higher-order Semantics (Session 21)

William Schuler

So far...

Last time:

1. represent meaning of partial predicates: ***lambda calculus***

So far...

Last time:

1. represent meaning of partial predicates: ***lambda calculus***
 - ***denotation*** = meaning of function: list associated inputs, outputs

So far...

Last time:

1. represent meaning of partial predicates: ***lambda calculus***
 - ***denotation*** = meaning of function: list associated inputs, outputs
 - ***interpretation*** = process of extracting meanings from symbols

So far...

Last time:

1. represent meaning of partial predicates: ***lambda calculus***
 - ***denotation*** = meaning of function: list associated inputs, outputs
 - ***interpretation*** = process of extracting meanings from symbols
 - ***1st- vs. 2nd-order*** denot. = set of consts vs. set of sets of const
 $\{C_1, C_2, C_3\}$ vs. $\{\{C_1, C_2\}, \{C_1, C_3\}, \{C_2, C_3\}\}$

So far...

Last time:

1. represent meaning of partial predicates: ***lambda calculus***
2. represent uncertainty in semantics:
stochastic lambda calculus (denotations \rightarrow conditional distribs)

So far...

Last time:

1. represent meaning of partial predicates: ***lambda calculus***
2. represent uncertainty in semantics:
stochastic lambda calculus (denotations \rightarrow conditional distribs)

$X = \llbracket \text{the TV is on} \rrbracket$ deterministic: $X = 1$, stochastic: $X =$

0	.67
1	.5

So far...

Last time:

1. represent meaning of partial predicates: ***lambda calculus***
2. represent uncertainty in semantics:
stochastic lambda calculus (denotations \rightarrow conditional distribs)

$$X = \llbracket \text{the TV is on} \rrbracket \text{ deterministic: } X = 1, \text{ stochastic: } X = \begin{array}{|c|c|} \hline 0 & .67 \\ \hline 1 & .5 \\ \hline \end{array}$$

$$X = \llbracket \text{the TV} \rrbracket \text{ deterministic: } X = C_2, \text{ stochastic: } X = \begin{array}{|c|c|} \hline C_1 & .5 \\ \hline C_2 & .3 \\ \hline \vdots & \vdots \\ \hline \end{array}$$

So far...

Last time:

1. represent meaning of partial predicates: ***lambda calculus***
2. represent uncertainty in semantics:
stochastic lambda calculus (denotations \rightarrow conditional distribs)
3. combine semantic uncertainty with syntactic/. . . uncertainty:
interleaved (statistical) interpretation – sem informs syn

So far...

Last time:

1. represent meaning of partial predicates: ***lambda calculus***
2. represent uncertainty in semantics:
stochastic lambda calculus (denotations \rightarrow conditional distribs)
3. combine semantic uncertainty with syntactic/. . . uncertainty:
interleaved (statistical) interpretation – sem informs syn

bottom-up backward distribs as in PCFG, but w. explicit denotations

$$P(\text{the TV is on, S:1}) = \sum \dots P(\text{S:1} \rightarrow \text{NP:C}_1 \text{ VP:}\langle \text{C}_1, 1 \rangle \mid \text{S:1}) \\ \dots \cdot P(\text{the TV} \mid \text{NP:C}_1) \cdot P(\text{is on} \mid \text{VP:}\langle \text{C}_1, 1 \rangle)$$

So far...

Last time:

1. represent meaning of partial predicates: ***lambda calculus***
2. represent uncertainty in semantics:
stochastic lambda calculus (denotations \rightarrow conditional distribs)
3. combine semantic uncertainty with syntactic/. . . uncertainty:
interleaved (statistical) interpretation – sem informs syn

bottom-up backward distribs as in PCFG, but w. explicit denotations

$$P(\text{the TV is on}, S:1) = \sum \dots \cdot P(\text{the TV} \mid NP:C_1) \cdot P(\text{is on} \mid VP:\langle C_1, 1 \rangle)$$

(requires ***isomorphic*** composition, first-order denotations)

So far...

Last time:

1. represent meaning of partial predicates: ***lambda calculus***
2. represent uncertainty in semantics:
stochastic lambda calculus (denotations \rightarrow conditional distribs)
3. combine semantic uncertainty with syntactic/. . . uncertainty:
interleaved (statistical) interpretation – sem informs syn

bottom-up backward distribs as in PCFG, but w. explicit denotations

$$P(\text{the TV is on, S:1}) = \sum \dots P(\text{S:1} \rightarrow \text{NP:C}_1 \text{ VP:}\langle \text{C}_1, 1 \rangle \mid \text{S:1}) \cdot P(\text{the TV} \mid \text{NP:C}_1) \cdot P(\text{is on} \mid \text{VP:}\langle \text{C}_1, 1 \rangle)$$

(requires ***isomorphic*** composition, first-order denotations)

Today: Problems for first-order denotations – quantifiers

Problems with quantifiers

Quantifiers don't have isomorphic composition, 1st-order denotations:

the TV is on: $On(C_1)$

every TV is on / most TVs are on: $On(??)$

Problems with quantifiers

Quantifiers don't have isomorphic composition, 1st-order denotations:

the TV is on: $On(C_1)$

every TV is on / most TVs are on: $On(??)$

$$\forall x. TV(x) \rightarrow On(x)$$

Problems with quantifiers

Quantifiers don't have isomorphic composition, 1st-order denotations:

the TV is on: $On(C_1)$

every TV is on / most TVs are on: $On(??)$

$$\forall x. TV(x) \rightarrow On(x)$$

Function assoc. w. 'every' takes predicate as arg: $\lambda R, S. R(x) \rightarrow S(x)$

Not isomorphic to syntax!

Problems with quantifiers

Quantifiers don't have isomorphic composition, 1st-order denotations:

the TV is on: $On(C_1)$

every TV is on / most TVs are on: $On(??)$

$$\forall x. TV(x) \rightarrow On(x)$$

Function assoc. w. 'every' takes predicate as arg: $\lambda R, S. R(x) \rightarrow S(x)$

Not isomorphic to syntax!

Can fix this with functions: $\lambda R. \lambda S. R(x) \rightarrow S(x)$

Problems with quantifiers

Quantifiers don't have isomorphic composition, 1st-order denotations:

the TV is on: $On(C_1)$

every TV is on / most TVs are on: $On(??)$

$$\forall x. TV(x) \rightarrow On(x)$$

Function assoc. w. 'every' takes predicate as arg: $\lambda R, S. R(x) \rightarrow S(x)$

Not isomorphic to syntax!

Can fix this with functions: $\lambda R. \lambda S. R(x) \rightarrow S(x)$

But now arguments are lambda exprs (denote sets!) – Not 1st-order!

Problems with quantifiers

Quantifiers don't have isomorphic composition, 1st-order denotations:

the TV is on: $On(C_1)$

every TV is on / most TVs are on: $On(??)$

$$\forall x. TV(x) \rightarrow On(x)$$

Function assoc. w. 'every' takes predicate as arg: $\lambda R, S. R(x) \rightarrow S(x)$

Not isomorphic to syntax!

Can fix this with functions: $\lambda R. \lambda S. R(x) \rightarrow S(x)$

But now arguments are lambda exprs (denote sets!) – Not 1st-order!

Worse: quantifiers can raise!

(In a control room) every TV gets one channel: $\forall t. \exists c. Receives(t, c)$

Problems with quantifiers

Quantifiers don't have isomorphic composition, 1st-order denotations:

the TV is on: $On(C_1)$

every TV is on / most TVs are on: $On(??)$

$$\forall x. TV(x) \rightarrow On(x)$$

Function assoc. w. 'every' takes predicate as arg: $\lambda R, S. R(x) \rightarrow S(x)$

Not isomorphic to syntax!

Can fix this with functions: $\lambda R. \lambda S. R(x) \rightarrow S(x)$

But now arguments are lambda exprs (denote sets!) – Not 1st-order!

Worse: quantifiers can raise!

(In a control room) every TV gets one channel: $\forall t. \exists c. Receives(t, c)$

(In my country. . .) every TV gets one channel: $\exists c. \forall t. Receives(t, c)$

Problems with quantifiers

Quantifiers don't have isomorphic composition, 1st-order denotations:

the TV is on: $On(C_1)$

every TV is on / most TVs are on: $On(??)$

$$\forall x. TV(x) \rightarrow On(x)$$

Function assoc. w. 'every' takes predicate as arg: $\lambda R, S. R(x) \rightarrow S(x)$

Not isomorphic to syntax!

Can fix this with functions: $\lambda R. \lambda S. R(x) \rightarrow S(x)$

But now arguments are lambda exprs (denote sets!) – Not 1st-order!

Worse: quantifiers can raise!

(In a control room) every TV gets one channel: $\forall t. \exists c. Receives(t, c)$

(In my country. . .) every TV gets one channel: $\exists c. \forall t. Receives(t, c)$

Quant. of inner NP outscopes quant. of outer NP! **Very non-isomorphic!**

Quasi-logical form

Process quantifiers bottom-up using quantifier storage:

(In my country) every TV gets one channel: $\exists c. \forall t. \textit{Receives}(t, c)$

Quasi-logical form

Process quantifiers bottom-up using quantifier storage:

(In my country) every TV gets one channel: $\exists c. \forall t. \text{Receives}(t, c)$

NPs, VPs compose normally (brackets designate quantifier storage):

Det:*EVERY* = $\lambda R, S. \forall x R(x) \rightarrow S(x)$ \rightarrow every

Det:*ONE* = $\lambda R, S. \exists x R(x) \wedge S(x)$ \rightarrow one

NP:[\$1\$2] \rightarrow Det:\$1 N:\$2

VP:\$1(\$2) \rightarrow V:\$1 NP:\$2

Quasi-logical form

Process quantifiers bottom-up using quantifier storage:

(In my country) every TV gets one channel: $\exists c. \forall t. \text{Receives}(t, c)$

NPs, VPs compose normally (brackets designate quantifier storage):

Det:*EVERY* = $\lambda R, S. \forall x R(x) \rightarrow S(x)$ \rightarrow every

Det:*ONE* = $\lambda R, S. \exists x R(x) \wedge S(x)$ \rightarrow one

NP:[\$1\$2] \rightarrow Det:\$1 N:\$2

VP:\$1(\$2) \rightarrow V:\$1 NP:\$2

But store (inside arguments) can be popped off:

VP:\$*Q*(\$*R*, \$*S*) \rightarrow VP:\$*S*([\$*Q*\$*R*])

VP:\$*S*[\$*Q*\$*R*] \rightarrow VP:\$*S*([\$*Q*\$*R*])

S:\$2[...][\$1] \rightarrow NP:\$1 VP:\$2[...]

S:\$*Q*(\$*R*, \$1)[...][...] \rightarrow S:\$1[...][\$*Q*\$*R*][...]

Quasi-logical form

Process quantifiers bottom-up using quantifier storage:

(In my country) every TV gets one channel: $\exists c. \forall t. \text{Receives}(t, c)$

NPs, VPs compose normally (brackets designate quantifier storage):

Det:*EVERY* = $\lambda R, S. \forall x R(x) \rightarrow S(x)$ \rightarrow every

Det:*ONE* = $\lambda R, S. \exists x R(x) \wedge S(x)$ \rightarrow one

NP:[\$1\$2] \rightarrow Det:\$1 N:\$2

VP:\$1(\$2) \rightarrow V:\$1 NP:\$2

But store (inside arguments) can be popped off:

VP:\$*Q*(\$*R*, \$*S*) \rightarrow VP:\$*S*([\$*Q*\$*R*])

VP:\$*S*[\$*Q*\$*R*] \rightarrow VP:\$*S*([\$*Q*\$*R*])

S:\$2[...][\$1] \rightarrow NP:\$1 VP:\$2[...]

S:\$*Q*(\$*R*, \$1)[...][...] \rightarrow S:\$1[...][\$*Q*\$*R*][...]

Store can have multi elements

Quasi-logical form

Process quantifiers bottom-up using quantifier storage:

(In my country) every TV gets one channel: $\exists c. \forall t. \text{Receives}(t, c)$

NPs, VPs compose normally (brackets designate quantifier storage):

Det:*EVERY* = $\lambda R, S. \forall x R(x) \rightarrow S(x)$ \rightarrow every

Det:*ONE* = $\lambda R, S. \exists x R(x) \wedge S(x)$ \rightarrow one

NP:[\$1\$2] \rightarrow Det:\$1 N:\$2

VP:\$1(\$2) \rightarrow V:\$1 NP:\$2

But store (inside arguments) can be popped off:

VP:\$Q(\$R, \$S) \rightarrow VP:\$S([\$Q\$R])

VP:\$S[\$Q\$R] \rightarrow VP:\$S([\$Q\$R])

S:\$2[...][\$1] \rightarrow NP:\$1 VP:\$2[...]

S:\$Q(\$R, \$1)[...][...] \rightarrow S:\$1[...][\$Q\$R][...]

Store can have multi elements – but unbounded stack: non-poly recog
