A Translation from Logic to English with Dynamic Semantics

Elizabeth Coppock and David Baxter

Cycorp Inc.

Logic and Engineering in Natural Language Semantics VI
Tokyo, Japan

General program

Goal: To define and implement a translation G from standard predicate logic (PL) into English text.

- Well, CycL (Lenat 1995) rather than PL, technically.
  - But CycL subsumes PL (Ramachandran et al. 2005).
- Not every possible output (cf. *completeness*); but every output should faithful to input (cf. *soundness*).
  - *Faithfulness:* If $\phi \rightarrow \psi$ then $G(\phi)$ "implies" $G(\psi)$.
- 'Inverse semantics'?

General program

Goal: To define and implement a translation G from standard predicate logic (PL) into English text.

- Well, CycL (Lenat 1995) rather than PL, technically.
  - But CycL subsumes PL (Ramachandran et al. 2005).
- Not every possible output (cf. *completeness*); but every output should faithful to input (cf. *soundness*).
  - *Faithfulness:* If $\phi \rightarrow \psi$ then $G(\phi)$ "implies" $G(\psi)$.
- 'Inverse semantics'?

General program

Goal: To define and implement a translation G from standard predicate logic (PL) into English text.

- Well, CycL (Lenat 1995) rather than PL, technically.
  - But CycL subsumes PL (Ramachandran et al. 2005).
- Not every possible output (cf. *completeness*); but every output should faithful to input (cf. *soundness*).
  - *Faithfulness:* If $\phi \rightarrow \psi$ then $G(\phi)$ "implies" $G(\psi)$.
- 'Inverse semantics'?

General program

Goal: To define and implement a translation G from standard predicate logic (PL) into English text.

- Well, CycL (Lenat 1995) rather than PL, technically.
  - But CycL subsumes PL (Ramachandran et al. 2005).
- Not every possible output (cf. *completeness*); but every output should faithful to input (cf. *soundness*).
  - *Faithfulness:* If $\phi \rightarrow \psi$ then $G(\phi)$ "implies" $G(\psi)$.
- 'Inverse semantics'?

Referring and non-referring expressions

Translating logical formulas into natural language requires generating both:

- *referring expressions*
  $\sim$ constants or closed non-atomic terms
- *non-referring expressions*
  $\sim$ variables

Karttunen's *discourse referents*

- Karttunen (1976): "the appearance of an indefinite noun phrase establishes a *discourse referent* just in case it justifies the occurrence of a coreferential pronoun or a definite noun phrase later in the text."
- This definition allows the study of coreference to proceed "independently of any general theory of extralinguistic reference" (p. 367).

Karttunen's *discourse referents*

- Karttunen (1976): "the appearance of an indefinite noun phrase establishes a *discourse referent* just in case it justifies the occurrence of a coreferential pronoun or a definite noun phrase later in the text."
- This definition allows the study of coreference to proceed "independently of any general theory of extralinguistic reference" (p. 367).

Short term referents

A normal referent survives for a while:

I found Mary's cat and kept it. Then it ran away.

A short term referent can die abruptly (Heim 1983):

*Everybody found a cat and kept it. *Then it ran away.*

Short term referents

A normal referent survives for a while:

I found Mary's cat and kept it. Then it ran away.

A short term referent can die abruptly (Heim 1983):

*Everybody found a cat and kept it. *Then it ran away.*

Short term referents

A normal referent survives for a while:

> *I found Mary's cat and kept it. Then it ran away.*

A short term referent can die abruptly (Heim 1983):

> *Everybody found a cat and kept it. *Then it ran away.*

Short term referents

A normal referent survives for a while:

I found Mary's cat and kept it. Then it ran away.

A short term referent can die abruptly (Heim 1983):

*Everybody found a cat and kept it. *Then it ran away.*

Introducing short-term referents

Short-term discourse referents are generally introduced with one of the following determiner series:

every	*everyone, everything, everywhere, every man, ...*
some	*someone, something, somewhere, some man, ...*
any	*anyone, anything, anywhere, any man, ...*
no	*noone, nothing, nowhere, no man, ...*

Universal quantification $\rightsquigarrow$ *every*

$\forall x[\textbf{loves}(\textbf{Mary}, x)]$

$\rightsquigarrow$

Mary loves everything.

Universal quantification + negation $\rightsquigarrow$ *no/any*

$\forall x[\neg \mathbf{loves}(\mathbf{Mary}, x)]$

$\rightsquigarrow$

Mary loves nothing.

Mary doesn't love anything.

Relative scope also important

$\neg\forall x[\textbf{loves}(\textbf{Mary}, x)]$

$\rightsquigarrow$

Mary doesn't love everything.

$\forall x[\neg\textbf{loves}(\textbf{Mary}, x)]$

$\rightsquigarrow$

Mary doesn't love anything.

Existentials $\rightsquigarrow$ indefinite

$\exists x[\textbf{loves}(\textbf{Doug}, x)]$

$\rightsquigarrow$

Doug loves something.

Conditional protasis $\rightsquigarrow$ indefinite

$\forall x[\mathbf{loves}(\mathbf{Doug}, x) \rightarrow \mathbf{loves}(\mathbf{Mary}, x)]$

$\rightsquigarrow$

If Doug loves something, then Mary loves it.

Different determiners, different lifespans

Referents introduced by indefinites last longer:

If a farmer owns $\left\{ \begin{array}{c} a_i \textit{ donkey} \\ *\textit{every}_i \textit{ donkey} \end{array} \right\}$*, then he beats* $\textit{it}_i$.

Goals of present work

- Enforce lifespan limitations for short-term discourse referents
- Introduce short-term discourse referents appropriately

Goals of present work

- Enforce lifespan limitations for short-term discourse referents
- Introduce short-term discourse referents appropriately

Outline

Related strands of research

Many areas contribute to a solution:

- (tactical) natural language generation
- generating referring expressions
- coreference
- truth-conditional semantics
- dynamic semantics

None solves the problem completely.

Natural language generation

- NL generation can be ‘tactical’ or ‘strategic’; this is tactical.
- Other tactical sentence generators exist.
- Do not deal with (non-)referring expressions.

Unification Categorial Grammar: Calder et al. (1989); LFG Wedekind and Kaplan (1996); Wedekind (1999); HPSG: Wilcock and Matsumoto (1998); Carroll et al. (1990); TAG: Becker (1998); Semantic head-driven generation: Shieber (1990); Van Noord (1994); Wilcock and Matsumoto (1998); Kikui (1992); Tuells et al. (2008)

Generating referring expressions

- Multiple books, one on table $\rightsquigarrow$ *the book on the table*
- Does not address non-referring expressions.

Horacek (1988); Dale (1989); Reiter and Dale (1992); Dale and Reiter (1994); Copestake et al. (1995); Shaw and McKeown (2000); Krahmer et al. (2003); Van Deemter (2002); Siddharthan and Copestake (2004); Varges and Deemter (2005); Kelleher and Kruijff (2006); Paraboni et al. (2007); Areces et al. (2008)

Theories of coreference

- Give models of acceptable pronoun use
- Applicable to referring and non-referring expressions.
- But do not deal with lifespan limitations or how to introduce short-term discourse referents.

Grosz et al. (1983); Gundel et al. (1993); Brennan (1995); Grosz et al. (1995); Beaver (2004)...

Truth-conditional semantics

- Deals with distribution of *every*, *no*, *any*, and *some*
- Does not model changes that occur as a discourse proceeds.

Montague (1974); Ladusaw (1980); Linebarger (1981); De Swart (1998); Giannakidou (1998, 2002); Szabolcsi (2004)...

Dynamic Semantics

- Dynamic semantics is designed for non-referring expressions.
- DRSs of DRT are built up during *interpretation*.
- Dynamic Montague Grammar (Groenendijk and Stokhof 1990), Compositional DRT (Muskens 1996), Van Leusen and Muskens (2003) are declaratively stated; use them?
  - No; it is not standard predicate logic into which these theories translate natural language.

Heim (1982); Kamp and Reyle (1993); Groenendijk and Stokhof (1990, 1991); Muskens (1996); Groenendijk et al. (1996); Van Leusen and Muskens (2003)...

Outline

The Cyc NL generation system

- Generates an English string, given logical input
- Captures lifespan limitations for discourse referents
- Introduces short-term discourse referents appropriately

Side effects

- We recursively define a generation function $G(\alpha)$, where α can be any expression of the logic
- G depends not only on α, but also on the discourse context D and the operator context O
- G can modify D and O

Side effects

- We recursively define a generation function $G(\alpha)$, where α can be any expression of the logic
- G depends not only on α, but also on the discourse context D and the operator context O
- G can modify D and O

Side effects

- We recursively define a generation function $G(\alpha)$, where α can be any expression of the logic
- G depends not only on α, but also on the discourse context D and the operator context O
- G can modify D and O

Discourse context

A discourse context D contains a set of *discourse referents*, composed of:

- a logical expression α, either an individual-denoting closed term or a variable ranging over individuals.
- *index features*: person, number and gender
  - These will not be shown.

Letting discourse referents be variables or constants

DRT treats all discourse referents as variables. Issues:

- Conversion to PL is complicated (Kamp and Reyle 1993):

x, y
$\mathbf{Mary}(x)$
$\mathbf{dog}(y)$
$\mathbf{owns}(x, y)$

$$\rightsquigarrow \exists xy[x = \mathbf{Mary} \wedge \mathbf{dog}(y) \wedge \mathbf{owns}(x, y)]$$

- Allowing constants to be discourse referents eliminates the need for 'external anchoring' (Muskens 1996).
- Allowing constant discourse referents is natural from NL generation perspective.

Letting discourse referents be variables or constants

DRT treats all discourse referents as variables. Issues:

- Conversion to PL is complicated (Kamp and Reyle 1993):

x, y
$\mathbf{Mary}(x)$
$\mathbf{dog}(y)$
$\mathbf{owns}(x, y)$

$$\rightsquigarrow \exists xy[x = \mathbf{Mary} \wedge \mathbf{dog}(y) \wedge \mathbf{owns}(x, y)]$$

- Allowing constants to be discourse referents eliminates the need for 'external anchoring' (Muskens 1996).
- Allowing constant discourse referents is natural from NL generation perspective.

Letting discourse referents be variables or constants

DRT treats all discourse referents as variables. Issues:

- Conversion to PL is complicated (Kamp and Reyle 1993):

x, y
$\mathbf{Mary}(x)$
$\mathbf{dog}(y)$
$\mathbf{owns}(x, y)$

$$\rightsquigarrow \exists xy[x = \mathbf{Mary} \wedge \mathbf{dog}(y) \wedge \mathbf{owns}(x, y)]$$

- Allowing constants to be discourse referents eliminates the need for 'external anchoring' (Muskens 1996).
- Allowing constant discourse referents is natural from NL generation perspective.

An example syntax tree

$$\begin{bmatrix} \text{PHON} & \text{“Mary loves herself”} \\ \text{SEM} & \textbf{loves}(\textbf{Mary}, \textbf{Mary}) \end{bmatrix}$$

$$\begin{bmatrix} \text{PHON} & \text{“Mary”} \\ \text{SEM} & \textbf{Mary} \end{bmatrix} \quad \begin{bmatrix} \text{PHON} & \text{“loves”} \\ \text{SEM} & \textbf{loves} \end{bmatrix} \quad \begin{bmatrix} \text{PHON} & \text{“herself”} \\ \text{SEM} & \textbf{Mary} \end{bmatrix}$$

D: {Mary}

An example syntax tree

$$\begin{bmatrix} \text{PHON} & \text{“Mary loves herself”} \\ \text{SEM} & \textbf{loves}(\textbf{Mary}, \textbf{Mary}) \end{bmatrix}$$

$$\begin{bmatrix} \text{PHON} & \text{“Mary”} \\ \text{SEM} & \textbf{Mary} \end{bmatrix} \quad \begin{bmatrix} \text{PHON} & \text{“loves”} \\ \text{SEM} & \textbf{loves} \end{bmatrix} \quad \begin{bmatrix} \text{PHON} & \text{“herself”} \\ \text{SEM} & \textbf{Mary} \end{bmatrix}$$

D: {Mary}

An example syntax tree

$$\begin{bmatrix} \text{PHON} & \text{“Mary loves herself”} \\ \text{SEM} & \textbf{loves}(\textbf{Mary}, \textbf{Mary}) \end{bmatrix}$$

$$\begin{bmatrix} \text{PHON} & \text{“Mary”} \\ \text{SEM} & \textbf{Mary} \end{bmatrix} \quad \begin{bmatrix} \text{PHON} & \text{“loves”} \\ \text{SEM} & \textbf{loves} \end{bmatrix} \quad \begin{bmatrix} \text{PHON} & \text{“herself”} \\ \text{SEM} & \textbf{Mary} \end{bmatrix}$$

D: $\{\textbf{Mary}\}$

An example syntax tree

$$\begin{bmatrix} \text{PHON} & \text{“Mary loves herself”} \\ \text{SEM} & \textbf{loves}(\textbf{Mary}, \textbf{Mary}) \end{bmatrix}$$

$$\begin{bmatrix} \text{PHON} & \text{“Mary”} \\ \text{SEM} & \textbf{Mary} \end{bmatrix} \quad \begin{bmatrix} \text{PHON} & \text{“loves”} \\ \text{SEM} & \textbf{loves} \end{bmatrix} \quad \begin{bmatrix} \text{PHON} & \text{“herself”} \\ \text{SEM} & \textbf{Mary} \end{bmatrix}$$

D: $\{\textbf{Mary}\}$

An example syntax tree

$$\begin{bmatrix} \text{PHON} & \text{“Mary loves herself”} \\ \text{SEM} & \textbf{loves}(\textbf{Mary}, \textbf{Mary}) \end{bmatrix}$$

$$\begin{bmatrix} \text{PHON} & \text{“Mary”} \\ \text{SEM} & \textbf{Mary} \end{bmatrix} \quad \begin{bmatrix} \text{PHON} & \text{“loves”} \\ \text{SEM} & \textbf{loves} \end{bmatrix} \quad \begin{bmatrix} \text{PHON} & \text{“herself”} \\ \text{SEM} & \textbf{Mary} \end{bmatrix}$$

D: $\{\textbf{Mary}\}$

An example syntax tree

$$\begin{bmatrix} \text{PHON} & \text{“Mary loves herself”} \\ \text{SEM} & \textbf{loves}(\textbf{Mary}, \textbf{Mary}) \end{bmatrix}$$

$$\begin{bmatrix} \text{PHON} & \text{“Mary”} \\ \text{SEM} & \textbf{Mary} \end{bmatrix} \quad \begin{bmatrix} \text{PHON} & \text{“loves”} \\ \text{SEM} & \textbf{loves} \end{bmatrix} \quad \begin{bmatrix} \text{PHON} & \text{“herself”} \\ \text{SEM} & \textbf{Mary} \end{bmatrix}$$

D: $\{\textbf{Mary}\}$

Lifespan limitations

If α is a quantificational sentence binding a variable ξ, then ξ must be removed from D after computing $G(\alpha)$.

Lifespan limitations

D: {**Doug**, x, **Mary**}

$$\forall x \neg [\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Doug loves nothing and Mary loves everything

Lifespan limitations

D: {Doug, x, Mary}

$\forall x \neg[\textbf{loves}(\textbf{Doug}, x)] \wedge \forall x[\textbf{loves}(\textbf{Mary}, x)]$

$\rightsquigarrow$

Doug loves nothing and Mary loves everything

Lifespan limitations

D: {**Doug**, x, **Mary**}

$$\forall x \neg [\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x [\mathbf{loves}(\mathbf{Mary}, x)]$$

$\rightsquigarrow$

Doug loves nothing and Mary loves everything

Lifespan limitations

D: {**Doug**, x, **Mary**}

$\forall x \neg[\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x[\mathbf{loves}(\mathbf{Mary}, x)]$

$\rightsquigarrow$

Doug loves nothing and Mary loves everything

Lifespan limitations

D: {**Doug**, x, **Mary**}

$$\forall x \neg [\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Doug loves nothing and Mary loves everything

Lifespan limitations

D: {**Doug**, x, Mary}

$$\forall x \neg [\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x [\mathbf{loves}(\mathbf{Mary}, x)]$$

$\rightsquigarrow$

Doug loves nothing and Mary loves everything

Lifespan limitations

D: {**Doug**, x, **Mary**}

$\forall x \neg[\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x[\mathbf{loves}(\mathbf{Mary}, x)]$

$\rightsquigarrow$

Doug loves nothing and Mary loves everything

Lifespan limitations

D: {**Doug**, x, **Mary**}

$$\forall x \neg [\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Doug loves nothing and Mary loves everything

Lifespan limitations

D: {**Doug**, x, **Mary**}

$$\forall x \neg [\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x [\mathbf{loves}(\mathbf{Mary}, x)]$$

$\rightsquigarrow$

Doug loves nothing and Mary loves everything

Lifespan limitations

D: {**Doug**, x, **Mary**}

$$\forall x \neg [\textbf{loves}(\textbf{Doug}, x)] \wedge \forall x [\textbf{loves}(\textbf{Mary}, x)]$$

$\rightsquigarrow$

Doug loves nothing and Mary loves everything

Lifespan limitations

D: {**Doug**, x, **Mary**}

$$\forall x \neg [\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Doug loves nothing and Mary loves everything

Lifespan limitations

D: $\{\mathbf{Doug}, x, \mathbf{Mary}\}$

$$\forall x \neg[\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x[\mathbf{loves}(\mathbf{Mary}, x)]$$

$\rightsquigarrow$

Doug loves nothing and Mary loves everything

Suppose we did not remove x from D...

D: {**Doug**, x, **Mary**}

$$\forall x \neg [\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Doug loves nothing and Mary loves it

Suppose we did not remove x from D...

D: {**Doug**, x, **Mary**}

$$\forall x \neg [\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Doug loves nothing and Mary loves it

Suppose we did not remove x from D...

D: {**Doug**, x, **Mary**}

$$\forall x \neg [\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Doug loves nothing and Mary loves it

Suppose we did not remove x from D...

D: {**Doug**, x, **Mary**}

$$\forall x \neg [\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Doug loves nothing and Mary loves it

Suppose we did not remove x from D...

D: {**Doug**, x, **Mary**}

$$\forall x \neg [\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Doug loves nothing and Mary loves it

Suppose we did not remove x from D...

D: $\{\mathbf{Doug}, x, \mathbf{Mary}\}$

$$\forall x \neg[\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x[\mathbf{loves}(\mathbf{Mary}, x)]$$

$\rightsquigarrow$

Doug loves nothing and Mary loves it

Suppose we did not remove x from D...

D: $\{\mathbf{Doug}, x, \mathbf{Mary}\}$

$$\forall x \neg[\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x[\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Doug loves nothing and Mary loves it

Suppose we did not remove x from D...

D: $\{\mathbf{Doug}, x, \mathbf{Mary}\}$

$$\forall x \neg[\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x[\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Doug loves nothing and Mary loves it

Suppose we did not remove x from D...

D: $\{\mathbf{Doug}, x, \mathbf{Mary}\}$

$$\forall x \neg[\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x[\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Doug loves nothing and Mary loves it

Suppose we did not remove x from D...

D: $\{\mathbf{Doug}, x, \mathbf{Mary}\}$

$$\forall x \neg[\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x[\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Doug loves nothing and Mary loves it

Suppose we did not remove x from D...

D: $\{\mathbf{Doug}, x, \mathbf{Mary}\}$

$$\forall x \neg[\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x[\mathbf{loves}(\mathbf{Mary}, x)]$$

$\rightsquigarrow$

Doug loves nothing and Mary loves it

Suppose we did not remove x from D...

D: $\{\mathbf{Doug}, x, \mathbf{Mary}\}$

$$\forall x \neg[\mathbf{loves}(\mathbf{Doug}, x)] \wedge \forall x[\mathbf{loves}(\mathbf{Mary}, x)]$$

$\rightsquigarrow$

Doug loves nothing and Mary loves it ← *bad!*

Returning to Heim's cat example

$\forall x \exists y[\textbf{found}(x, y) \wedge \textbf{kept}(x, y)]$

$\rightsquigarrow$

Everybody found a cat and kept it$_i$.

Now, x and y are removed from the discourse context.

It$_i$ ran away.

Returning to Heim's cat example

$\forall x \exists y[\textbf{found}(x, y) \wedge \textbf{kept}(x, y)]$

$\rightsquigarrow$

Everybody found a cat and kept it$_i$.

Now, x and y are removed from the discourse context.

It$_i$ ran away.

Returning to Heim's cat example

$\forall x \exists y[\textbf{found}(x, y) \wedge \textbf{kept}(x, y)]$

$\rightsquigarrow$

Everybody found a cat and kept it$_i$.

Now, x and y are removed from the discourse context.

It$_i$ ran away.

Returning to Heim's cat example

$\forall x \exists y[\textbf{found}(x, y) \wedge \textbf{kept}(x, y)]$

$\rightsquigarrow$

Everybody found a cat and kept it$_i$.

Now, x and y are removed from the discourse context.

It$_i$ ran away. $\leftarrow$ *impossible*

Operator Context: Definition

We define an *operator context* O as a tuple $\langle V, S, n \rangle$ where:

- V is a set of variable type entries
- S is a stack of logical symbols
- n is an integer representing the number of negations remaining to be expressed.

A variable type entry $v = \langle \alpha, \theta, \tau \rangle$, where:

- α is a variable over individuals,
- θ is a quantifier symbol
- τ is a type

Operator Context: Definition

We define an *operator context* O as a tuple $\langle V, S, n \rangle$ where:

- V is a set of variable type entries
- S is a stack of logical symbols
- n is an integer representing the number of negations remaining to be expressed.

A variable type entry $v = \langle \alpha, \theta, \tau \rangle$, where:

- α is a variable over individuals,
- θ is a quantifier symbol
- τ is a type

Constructing a clausal skeleton

V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Constructing a clausal skeleton

V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Constructing a clausal skeleton

V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Clausal skeletons: Eliminating the antecedent

V: $\{\langle x, \forall, \mathbf{Man}\rangle\}$

$$\mathbf{isa}(x, \mathbf{Man}) \rightarrow \mathbf{loves}(x, \mathbf{Mary})$$

$\rightsquigarrow$

Every man loves Mary.

Clausal skeletons: Eliminating the antecedent

V: $\{\langle x, \forall, \mathbf{Man}\rangle\}$

$\mathbf{isa}(x, \mathrm{Man}) \rightarrow \mathbf{loves}(x, \mathbf{Mary})$

$\rightsquigarrow$

Every man loves Mary.

Clausal skeletons: Eliminating the antecedent

V: $\{\langle x, \forall, \mathbf{Man}\rangle\}$

$$\text{isa}(x, \text{Man}) \rightarrow \mathbf{loves}(x, \mathbf{Mary})$$

$\rightsquigarrow$

Every man loves Mary.

Negation stripping

- When α is of the form $\neg\phi$, the clausal skeleton of α is ϕ.
- No variable type entries are produced in this case.
- But the counter representing the number of unexpressed negations, n, is incremented.
- And $\neg$ is pushed onto the operator stack.

Negation stripping

- When α is of the form $\neg\phi$, the clausal skeleton of α is ϕ.
- No variable type entries are produced in this case.
- But the counter representing the number of unexpressed negations, n, is incremented.
- And $\neg$ is pushed onto the operator stack.

Negation stripping

- When α is of the form $\neg\phi$, the clausal skeleton of α is ϕ.
- No variable type entries are produced in this case.
- But the counter representing the number of unexpressed negations, n, is incremented.
- And $\neg$ is pushed onto the operator stack.

Negation stripping

- When α is of the form $\neg\phi$, the clausal skeleton of α is ϕ.
- No variable type entries are produced in this case.
- But the counter representing the number of unexpressed negations, n, is incremented.
- And $\neg$ is pushed onto the operator stack.

Updating the operator stack

- If $\alpha \sim \forall \xi \phi$, then push ξ onto S and pop it off after $G(\alpha)$.
- If $\alpha \sim \neg \phi$ then push $\neg$ onto S and pop it off after $G(\alpha)$.
- If $\alpha \sim \phi \rightarrow \psi$ then push $\rightarrow$ onto S and pop it off after $G(\phi)$.

Realizing a variable

Suppose $\langle \xi, \theta, \tau \rangle \in V$.

- If ξ is in D, then realize it with a pronoun if its antecedent is sufficiently salient and the pronoun would not be ambiguous.
- Otherwise, realize it with a determiner followed by a noun expressing τ.

Determiner selection algorithm: Computing π

First step in choosing a determiner: Compute π.

- If $\theta = \forall$, is the variable is deeper on the stack than an NPI licenser ($\rightarrow$ or $\neg$)?
  - If so, set π equal to the NPI licenser
- If $\theta = \exists$, is there an NPI licenser is deeper than it?
  - If so, set π equal to the NPI licenser

If the variable has no NPI licenser, then π is null.

Determiner selection algorithm: Computing π

First step in choosing a determiner: Compute π.

- If $\theta = \forall$, is the variable is deeper on the stack than an NPI licenser ($\rightarrow$ or $\neg$)?
  - If so, set π equal to the NPI licenser
- If $\theta = \exists$, is there an NPI licenser is deeper than it?
  - If so, set π equal to the NPI licenser

If the variable has no NPI licenser, then π is null.

Determiner selection algorithm

Given π,

- If ξ is in D, return *that* or *the*.
- If π is non-null:
  - If $\pi = \neg$ and $n > 0$, then return *no* and decrement n by one.
  - Otherwise, return *any* or *a/an*.*
- If $\theta = \forall$, return *every*.
- Otherwise, return *a/an*.

Example: *nothing*

D: $\langle \mathbf{Mary}, x \rangle$
V: $\{\langle x, \forall, \mathbf{Thing} \rangle\}$
S: $\langle x, \neg \rangle$
n: 0

$$\forall x \neg [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Mary loves nothing

Example: *nothing*

D: $\langle \mathbf{Mary}, x \rangle$
V: $\{\langle x, \forall, \mathbf{Thing} \rangle\}$
S: $\langle x, \neg \rangle$
n: 0

$$\forall x \neg [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Mary loves nothing

Example: *nothing*

D: $\langle \mathbf{Mary}, x \rangle$
V: $\{\langle x, \forall, \mathbf{Thing} \rangle\}$
S: $\langle x, \neg \rangle$
n: 1

$$\forall x \neg [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Mary loves nothing

Example: *nothing*

D: $\langle \mathbf{Mary}, x \rangle$
V: $\{\langle x, \forall, \mathbf{Thing} \rangle\}$
S: $\langle x, \neg \rangle$
n: 1

$\forall x \neg [\mathbf{loves}(\mathbf{Mary}, x)]$

$\rightsquigarrow$

Mary loves nothing

Example: *nothing*

D: $\langle \mathbf{Mary}, x \rangle$
V: $\{\langle x, \forall, \mathbf{Thing} \rangle\}$
S: $\langle x, \neg \rangle$
n: 1

$$\forall x \neg [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Mary loves nothing

Example: *nothing*

D: $\langle \textbf{Mary}, x \rangle$
V: $\{\langle x, \forall, \textbf{Thing} \rangle\}$
S: $\langle x, \neg \rangle$
n: 1

$$\forall x \neg [\textbf{loves}(\textbf{Mary}, x)]$$

$$\rightsquigarrow$$

Mary loves nothing

Example: *nothing*

D: $\langle$**Mary**, $x\rangle$
V: $\{\langle x, \forall,$ **Thing**$\rangle\}$
S: $\langle x, \neg\rangle$
n: 1

$\forall x \neg$[**loves**(**Mary**, x)]

$\rightsquigarrow$

Mary loves nothing

Example: *nothing*

D: $\langle$**Mary**, $x\rangle$
V: $\{\langle x, \forall,$ **Thing**$\rangle\}$
S: $\langle x, \neg\rangle$
n: 1
π : $\neg$

$$\forall x \neg [\textbf{loves}(\textbf{Mary}, x)]$$

$$\rightsquigarrow$$

Mary loves nothing

Example: *nothing*

D: $\langle \textbf{Mary}, x \rangle$
V: $\{\langle x, \forall, \textbf{Thing} \rangle\}$
S: $\langle x, \neg \rangle$
n: 0

$$\forall x \neg [\textbf{loves}(\textbf{Mary}, x)]$$

$$\rightsquigarrow$$

Mary loves nothing

Example: *anything*

D: $\langle \mathbf{Mary}, x \rangle$
V: $\{\langle x, \forall, \mathbf{Thing} \rangle\}$
S: $\langle x, \neg \rangle$
n: 0

$$\forall x \neg [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Mary doesn't love anything

Example: *anything*

D: $\langle \mathbf{Mary}, x \rangle$
V: $\{\langle x, \forall, \mathbf{Thing} \rangle\}$
S: $\langle x, \neg \rangle$
n: 0

$$\forall x \neg [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Mary doesn't love anything

Example: *anything*

D: $\langle \mathbf{Mary}, x \rangle$
V: $\{\langle x, \forall, \mathbf{Thing} \rangle\}$
S: $\langle x, \neg \rangle$
n: 1

$$\forall x \neg [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Mary doesn't love anything

Example: *anything*

D: $\langle \mathbf{Mary}, x \rangle$
V: $\{\langle x, \forall, \mathbf{Thing} \rangle\}$
S: $\langle x, \neg \rangle$
n: 1

$$\forall x \neg [\mathbf{loves}(\mathbf{Mary}, x)]$$

$$\rightsquigarrow$$

Mary doesn't love anything

Example: *anything*

D: $\langle \textbf{Mary}, x \rangle$
V: $\{\langle x, \forall, \textbf{Thing} \rangle\}$
S: $\langle x, \neg \rangle$
n: 1

$$\forall x \neg [\textbf{loves}(\textbf{Mary}, x)]$$

$$\rightsquigarrow$$

Mary doesn't love anything

Example: *anything*

D: $\langle$**Mary**, $x\rangle$
V: $\{\langle x, \forall,$ **Thing**$\rangle\}$
S: $\langle x, \neg\rangle$
n: 1

$\forall x \neg$[**loves**(**Mary**, x)]

$\rightsquigarrow$

Mary doesn't love anything

Example: *anything*

D: $\langle$**Mary**, $x\rangle$
V: $\{\langle x, \forall,$ **Thing**$\rangle\}$
S: $\langle x, \neg\rangle$
n: 1

$$\forall x \neg [\textbf{loves}(\textbf{Mary}, x)]$$

$$\rightsquigarrow$$

Mary doesn't love anything

Example: *anything*

D: $\langle$**Mary**, $x\rangle$
V: $\{\langle x, \forall,$ **Thing**$\rangle\}$
S: $\langle x, \neg\rangle$
n: 0

$$\forall x \neg [\textbf{loves}(\textbf{Mary}, x)]$$

$\rightsquigarrow$

Mary doesn't love anything

Example: *anything*

D: $\langle \textbf{Mary}, x \rangle$
V: $\{\langle x, \forall, \textbf{Thing} \rangle\}$
S: $\langle x, \neg \rangle$
n: 0
π : $\neg$

$$\forall x \neg [\textbf{loves}(\textbf{Mary}, x)]$$

$$\rightsquigarrow$$

Mary doesn't love anything

Example: *anything*

D: $\langle \textbf{Mary}, x \rangle$
V: $\{\langle x, \forall, \textbf{Thing} \rangle\}$
S: $\langle x, \neg \rangle$
n: 0

$$\forall x \neg [\textbf{loves}(\textbf{Mary}, x)]$$

$$\rightsquigarrow$$

Mary doesn't love anything

Example: Donkey sentence

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, y, \rightarrow\rangle$
n: 0

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Example: Donkey sentence

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, y, \rightarrow\rangle$
n: 0

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Example: Donkey sentence

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, y, \rightarrow \rangle$
n: 0

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Example: Donkey sentence

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, y, \rightarrow \rangle$
n: 0

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Example: Donkey sentence

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, y, \rightarrow \rangle$
n: 0
π :$\rightarrow$

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Example: Donkey sentence

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, y, \rightarrow \rangle$
n: 0

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Example: Donkey sentence

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, y, \rightarrow \rangle$
n: 0

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Example: Donkey sentence

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, y, \rightarrow \rangle$
n: 0
π :$\rightarrow$

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Example: Donkey sentence

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, y, \rightarrow \rangle$
n: 0

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Example: Donkey sentence

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, y, \rightarrow\rangle$
n: 0

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Example: Donkey sentence

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, y, \rightarrow\rangle$
n: 0

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Example: Donkey sentence

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, y, \rightarrow\rangle$
n: 0

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Example: Donkey sentence

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, y, \rightarrow\rangle$
n: 0

$$\forall x \forall y[\textbf{isa}(x, \textbf{Farmer}) \wedge \textbf{isa}(y, \textbf{Donkey}) \wedge \textbf{owns}(x, y) \rightarrow \textbf{beats}(x, y)]$$

$\rightsquigarrow$

If a farmer owns a donkey, then he beats it.

Interaction between determiners and lifespans

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, \rightarrow, y\rangle$
n: 0

$$\forall x[\textbf{isa}(x, \textbf{Farmer}) \wedge \forall y[\textbf{isa}(y, \textbf{Donkey}) \rightarrow \textbf{owns}(x, y)]] \rightarrow \ldots$$

$\rightsquigarrow$

If a farmer owns every donkey, then . . .

Interaction between determiners and lifespans

D: $\{x, y\}$
V: $\{\langle x, \forall, \mathbf{Farmer}\rangle, \langle y, \forall, \mathbf{Donkey}\rangle\}$
S: $\langle x, \rightarrow, y\rangle$
n: 0

$$\forall x[\mathbf{isa}(x, \mathbf{Farmer}) \wedge \forall y[\mathbf{isa}(y, \mathbf{Donkey}) \rightarrow \mathbf{owns}(x, y)]] \rightarrow \ldots$$

$\rightsquigarrow$

If a farmer owns every donkey, then . . .

Interaction between determiners and lifespans

D: $\{x, y\}$
V: $\{\langle x, \forall, \mathbf{Farmer}\rangle, \langle y, \forall, \mathbf{Donkey}\rangle\}$
S: $\langle x, \rightarrow, y\rangle$
n: 0

$$\forall x[\mathbf{isa}(x, \mathbf{Farmer}) \wedge \forall y[\mathbf{isa}(y, \mathbf{Donkey}) \rightarrow \mathbf{owns}(x, y)]] \rightarrow \ldots$$

$\rightsquigarrow$

If a farmer owns every donkey, then . . .

Interaction between determiners and lifespans

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, \rightarrow, y\rangle$
n: 0

$$\forall x[\textbf{isa}(x, \textbf{Farmer}) \wedge \forall y[\textbf{isa}(y, \textbf{Donkey}) \rightarrow \textbf{owns}(x, y)]] \rightarrow \ldots$$

$\rightsquigarrow$

If a farmer owns every donkey, then . . .

Interaction between determiners and lifespans

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, \rightarrow, y\rangle$
n: 0

$$\forall x[\textbf{isa}(x, \textbf{Farmer}) \wedge \forall y[\textbf{isa}(y, \textbf{Donkey}) \rightarrow \textbf{owns}(x, y)]] \rightarrow \ldots$$

$\rightsquigarrow$

If a farmer owns every donkey, then . . .

Interaction between determiners and lifespans

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, \rightarrow, y\rangle$
n: 0

$$\forall x[\textbf{isa}(x, \textbf{Farmer}) \wedge \forall y[\textbf{isa}(y, \textbf{Donkey}) \rightarrow \textbf{owns}(x, y)]] \rightarrow \ldots$$

$\rightsquigarrow$

If a farmer owns every donkey, then . . .

Interaction between determiners and lifespans

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, \rightarrow, y\rangle$
n: 0

$$\forall x[\textbf{isa}(x, \textbf{Farmer}) \wedge \forall y[\textbf{isa}(y, \textbf{Donkey}) \rightarrow \textbf{owns}(x, y)]] \rightarrow \ldots$$

$\rightsquigarrow$

If a farmer owns every donkey, then . . .

Interaction between determiners and lifespans

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, \rightarrow, y\rangle$
n: 0

$$\forall x[\textbf{isa}(x, \textbf{Farmer}) \wedge \forall y[\textbf{isa}(y, \textbf{Donkey}) \rightarrow \textbf{owns}(x, y)]] \rightarrow \ldots$$

$\rightsquigarrow$

If a farmer owns every donkey, then . . .

Interaction between determiners and lifespans

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, \rightarrow, y\rangle$
n: 0
π : $\emptyset$

$$\forall x[\textbf{isa}(x, \textbf{Farmer}) \wedge \forall y[\textbf{isa}(y, \textbf{Donkey}) \rightarrow \textbf{owns}(x, y)]] \rightarrow \ldots$$

$\rightsquigarrow$

If a farmer owns every donkey, then . . .

Interaction between determiners and lifespans

D: $\{x, y\}$
V: $\{\langle x, \forall, \textbf{Farmer}\rangle, \langle y, \forall, \textbf{Donkey}\rangle\}$
S: $\langle x, \rightarrow, y\rangle$
n: 0

$$\forall x[\textbf{isa}(x, \textbf{Farmer}) \wedge \forall y[\textbf{isa}(y, \textbf{Donkey}) \rightarrow \textbf{owns}(x, y)]] \rightarrow \ldots$$

$\rightsquigarrow$

If a farmer owns every donkey, then . . .

Interaction between determiners and lifespans

D: $\{x, y\}$
V: $\{\langle x, \forall, \mathbf{Farmer}\rangle, \langle y, \forall, \mathbf{Donkey}\rangle\}$
S: $\langle x, \rightarrow, y\rangle$
n: 0

$$\forall x[\mathbf{isa}(x, \mathbf{Farmer}) \wedge \forall y[\mathbf{isa}(y, \mathbf{Donkey}) \rightarrow \mathbf{owns}(x, y)]] \rightarrow \ldots$$

$\rightsquigarrow$

If a farmer owns every donkey, then . . .

Interaction between determiners and lifespans

D: $\{x, y\}$
V: $\{\langle x, \forall, \mathbf{Farmer}\rangle, \langle y, \forall, \mathbf{Donkey}\rangle\}$
S: $\langle x, \rightarrow, y\rangle$
n: 0

$$\forall x[\mathbf{isa}(x, \mathbf{Farmer}) \wedge \forall y[\mathbf{isa}(y, \mathbf{Donkey}) \rightarrow \mathbf{owns}(x, y)]] \rightarrow \ldots$$

$\rightsquigarrow$

If a farmer owns every donkey, then . . .

Now y is not in $D \Rightarrow$ no anaphora to *every donkey*

Summary

- Translating predicate logic to English requires generating referring and non-referring expressions.
- Our solution uses a dynamically-updated *discourse context* D and *operator context* O.
  - D is used for referring and non-referring expressions
  - O is used for introducing non-referring expressions
- The system introduces non-referring expressions correctly, and enforces lifespan limitations on discourse referents.

Summary

- Translating predicate logic to English requires generating referring and non-referring expressions.
- Our solution uses a dynamically-updated *discourse context* D and *operator context* O.
  - D is used for referring and non-referring expressions
  - O is used for introducing non-referring expressions
- The system introduces non-referring expressions correctly, and enforces lifespan limitations on discourse referents.

Summary

- Translating predicate logic to English requires generating referring and non-referring expressions.
- Our solution uses a dynamically-updated *discourse context* D and *operator context* O.
  - D is used for referring and non-referring expressions
  - O is used for introducing non-referring expressions
- The system introduces non-referring expressions correctly, and enforces lifespan limitations on discourse referents.

Thank you!

Thanks to two anonymous reviewers, Lucas Champollion and Elias Ponvert, David Beaver, Nicholas Asher, and Cleo Condoravdi for feedback. This work was partially supported under the DARPA Rapid Knowledge Formation program.

Areces, C., Koller, A., and Striegnitz, K. (2008). Referring expressions as formulas of description logic. In White, M., Nakatsu, C., and McDonald, D., editors, *Proceedings of the Fifth International Natural Language Generation Conference*, pages 42–49, Salt Fork, Ohio. Association for Computational Linguistics.

Beaver, D. I. (2004). The optimization of discourse anaphora. *Linguistics and Philosophy*, 27:3–56.

Becker, T. (1998). Fully lexicalized head-driven syntactic generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, pages 208–217. Association for Computational Linguistics.

Brennan, S. (1995). Centering attention in discourse. *Language and Cognitive Processes*, 10:137–167.

Calder, J., Reape, M., and Zeevat, H. (1989). An algorithm for generation in unification categorial grammar. In *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics*, pages 233–240, Manchester, UK.

Carroll, J., Flickinger, D., Copestake, A., and Poznanski, V. (1990). An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation*, Toulouse, France.

Copestake, A., Flickinger, D., Malouf, R., Riehemann, S., and Sag, I. (1995). Translation using minimal recursion semantics. In *Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*, Leuven, Belgium.

Dale, R. (1989). Cooking up referring expressions. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*.

Dale, R. and Reiter, E. (1994). Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19:233–263.

De Swart, H. (1998). Licensing of negative polarity items under inverse scope. *Lingua*, 105:175–200.

Giannakidou, A. (1998). *Polarity Sensitivity as (Non)veridical Dependency*. John Benjamins, Amsterdam.

Giannakidou, A. (2002). Licensing and sensitivity in polarity items: from downward entailment to (non)veridicality. In Andronis, M., Pycha, A., and Yoshimura, K., editors, *CLS 38: Papers from the 38th Annual Meeting of the Chicago Linguistic Society, Parasession on Polarity and Negation*. Chicago Linguistic Society.

Groenendijk, J. and Stokhof, M. (1990). Dynamic Montague grammar. In *Proceedings of the Second Symposium on Logic and Language*, pages 3–48, Budapest.

Groenendijk, J. and Stokhof, M. (1991). Dynamic predicate logic. *Linguistics and Philosophy*, 14:39–100.

Groenendijk, J., Stokhof, M., and Veltman, F. (1996). Coreference and modality.

Grosz, B. J., Joshi, A. K., and Weinstein, S. (1983). Providing a unified account of definite noun phrases in discourse. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, pages 44–49, Cambridge, MA.

Grosz, B. J., Joshi, A. K., and Weinstein, S. (1995). Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21:203–226.

Gundel, J. K., Hedberg, N., and Zacharski, R. (1993). Cognitive status and the form of referring expressions in discourse. *Language*, 69:274–307.

Heim, I. (1982). *The Semantics of Definite and Indefinite Noun Phrases*. PhD thesis, MIT.

Heim, I. (1983). File change semantics and the familiarity theory of definiteness. In Baurle, R., Schwarze, C., and Von Stechow, A., editors, *Meaning, Use, and the Interpretation of Language*, pages 164–189. Walter de Gruyter, Berlin.

Horacek, H. (1988). An algorithm for generating referential descriptions with flexible interfaces. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 206–213.

Kamp, H. and Reyle, U. (1993). *From Discourse to Logic*. Kluwer Academic Publishers, Dordrecht.

Karttunen, L. (1976). Discourse referents. In McCawley, J. D., editor, *Syntax and Semantics 7: Notes from the Linguistic Underground*, pages 363–385. Academic Press, New York.

Kelleher, J. D. and Kruijff, G.-J. M. (2006). Incremental generation of spatial referring expressions in situated dialog. In *Proceedings of COLING/ACL-06*.

Kikui, G. (1992). Feature structure based semantic head driven generation. In *Proceedings of COLING-92*, pages 32–38, Nantes.

Krahmer, E., Van Erk, S., and Verleg, A. (2003). Graph-based generation of referring expressions. *Computational Linguistics*.

Ladusaw, W. A. (1980). *Polarity Sensitivity as Inherent Scope Relations*. Garland, New York.

Lenat, D. (1995). Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38.

Linebarger, M. C. (1981). *The grammar of negative polarity*. PhD thesis, MIT.

Montague, R. (1974). English as a formal language. In Thomason, R. H., editor, *Formal Philosophy: Selected Papers fo Richard Montague*, pages 188–221. Yale University Press, New Haven.

Muskens, R. (1996). Combining Montague semantics and discourse representation. *Linguistics and Philosophy*, 19:143–186.

Paraboni, I., Van Deemter, K., and Masthoff, J. (2007). Generating referring expressions: Making referents easy to identify. *Computational Linguistics*, 33:229–254.

Ramachandran, D., Reagan, P., and Goolsbey, K. (2005). First-orderized ResearchCyc: Expressivity and efficiency in a common-sense ontology. In *Papers from the AAAI Workshop on Contexts and Ontologies: Theory, Practice and Applications*, Pittsburg, PA.

Reiter, E. and Dale, R. (1992). A fast algorithm for the generation of referring expressions. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 232–238, Nantes.

Shaw, J. and McKeown, K. (2000). Generating referring quantified expressions. In *Proceedings of the first international conference on natural language generation*, pages 100–107, Mitzpe Ramon, Israel.

Shieber, S. M. (1990). Semantic head-driven generation. *Computational Linguistics*, 16:30–42.

Siddharthan, A. and Copestake, A. (2004). Generating referring expressions in open domains. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 408–415, Barcelona, Spain.

Szabolcsi, A. (2004). Positive polarity – negative polarity. *Natural Language and Linguistic Theory*, 22:409–452.

Tuells, T., Rigau, G., and Rodriguez, H. (2008). *Offline Compilation of Chains for Head-Driven Generation with Constraint-Based Grammars*, pages 267–286. Springer, Berlin.

Van Deemter, K. (2002). Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28:37–52.

Van Leusen, N. and Muskens, R. (2003). Construction by description in discourse representation. In Peregrin, J., editor, *Meaning: The Dynamic Turn*, pages 33–65, Oxford. Elsevier.

Van Noord, G. (1994). An overview of head-driven bottom-up generation. In Dale, R., Mellish, C., and Zock, M., editors, *Current Research in Natural Language Generation*, pages 141–165. Academic Press.

Varges, S. and Deemter, K. V. (2005). Generating referring expressions containing quantifiers. In *Proceedings of the 6th International Workshop on Computational Semantics*, pages 1–13.

Wedekind, J. (1999). Semantic-driven generation with LFG- and PATR-style grammars. *Computational Linguistics*, 25:277–281.

Wedekind, J. and Kaplan, R. M. (1996). Ambiguity-preserving generation with LFG- and PATR-style grammars. *Computational Linguistics*, 22:555–558.

Wilcock, G. and Matsumoto, Y. (1998). Head-driven generation with HPSG. In *Proceedings of COLING-ACL '98: Workshop on Usage of WordNet in Natural Language Processing Systems*, pages 1393–1397.