Semantics and Pragmatics of NLP
Lambda Terms, Quantifiers, Satisfaction

Alex Lascarides & Ewan Klein

School of Informatics
University of Edinburgh

10 January 2008

Outline

Transitive Verbs as Functions

We looked at replacing *n*-ary relations with functions. How does this work with transitive verbs?

- Version 1: chase of type $<\text{IND}, \text{IND}> \rightarrow \text{BOOL}$
- Version 2: chase of type $\text{IND} \rightarrow (\text{IND} \rightarrow \text{BOOL})$

Advantages of Version 2 (called a *curryed function*):

- Makes the syntax more uniform.
- Fits better with compositional semantics (discussed later)

Transitive Verbs as Functions

We looked at replacing *n*-ary relations with functions. How does this work with transitive verbs?

- Version 1: chase of type $<\text{IND}, \text{IND}> \rightarrow \text{BOOL}$
- Version 2: chase of type $\text{IND} \rightarrow (\text{IND} \rightarrow \text{BOOL})$

Advantages of Version 2 (called a *curryed function*):

- Makes the syntax more uniform.
- Fits better with compositional semantics (discussed later)

Transitive Verbs as Functions

We looked at replacing *n*-ary relations with functions. How does this work with transitive verbs?

- Version 1: chase of type $<\text{IND}, \text{IND}> \rightarrow \text{BOOL}$
- Version 2: chase of type $\text{IND} \rightarrow (\text{IND} \rightarrow \text{BOOL})$

Advantages of Version 2 (called a *curryed function*):

- Makes the syntax more uniform.
- Fits better with compositional semantics (discussed later)

Transitive Verbs as Functions

We looked at replacing *n*-ary relations with functions. How does this work with transitive verbs?

- Version 1: chase of type $<\text{IND}, \text{IND}> \rightarrow \text{BOOL}$
- Version 2: chase of type $\text{IND} \rightarrow (\text{IND} \rightarrow \text{BOOL})$

Advantages of Version 2 (called a *curryed function*):

- Makes the syntax more uniform.
- Fits better with compositional semantics (discussed later)

Transitive Verbs as Functions

We looked at replacing *n*-ary relations with functions. How does this work with transitive verbs?

- Version 1: chase of type $<\text{IND}, \text{IND}> \longrightarrow \text{BOOL}$
- Version 2: chase of type $\text{IND} \longrightarrow (\text{IND} \longrightarrow \text{BOOL})$

Advantages of Version 2 (called a *curryed function*):

- Makes the syntax more uniform.
- Fits better with compositional semantics (discussed later)

Lambda

Lambdas talk about missing information, and where it is.

- The λ binds a variable.
- The positions of a λ-bound variable in the formula mark where information is 'missing'.
- Replacing these variables with values fills in the missing information.

Example:

- $\lambda x.(\text{man}\ x)$ — λ-abstract
- $(\lambda x.(\text{man}\ x)\ \text{john})$ — application
- $(\text{man}\ \text{john})$ — β-reduction/function application.

Types

- IND and BOOL are basic types.
- If σ, τ are types, then so is $(\sigma \rightarrow \tau)$. Brackets are omitted if no ambiguity.
- For types τ, we have variables **Var**(τ), constants **Con**(τ).
- Since we are doing first order logic, we will later restrict variables to **Var**(IND), but allow constants of any type.

Types

- IND and BOOL are basic types.
- If σ, τ are types, then so is $(\sigma \rightarrow \tau)$. Brackets are omitted if no ambiguity.
- For types τ, we have variables **Var**(τ), constants **Con**(τ).
- Since we are doing first order logic, we will later restrict variables to **Var**(IND), but allow constants of any type.

Types

- IND and BOOL are basic types.
- If σ, τ are types, then so is $(\sigma \rightarrow \tau)$. Brackets are omitted if no ambiguity.
- For types τ, we have variables **Var**(τ), constants **Con**(τ).
- Since we are doing first order logic, we will later restrict variables to **Var**(IND), but allow constants of any type.

Types

- IND and BOOL are basic types.
- If σ, τ are types, then so is $(\sigma \rightarrow \tau)$. Brackets are omitted if no ambiguity.
- For types τ, we have variables **Var**(τ), constants **Con**(τ).
- Since we are doing first order logic, we will later restrict variables to **Var**(IND), but allow constants of any type.

Terms in Typed Lambda Calculus

We define terms **Term**(τ) of type τ:

- **Var**(τ) $\subseteq$ **Term**(τ).
- **Con**(τ) $\subseteq$ **Term**(τ).
- If $\alpha \in$ **Term**($\sigma \rightarrow \tau$) and $\beta \in$ **Term**(σ) then ($\alpha\,\beta$) $\in$ **Term**(τ) (function application).
- If $x \in$ **Var**(σ) and $\alpha \in$ **Term**(ρ), then $\lambda x.\alpha \in$ **Term**(τ), where $\tau = (\sigma \rightarrow \rho)$

Terms in Typed Lambda Calculus

We define terms **Term**(τ) of type τ:

- **Var**(τ) $\subseteq$ **Term**(τ).
- **Con**(τ) $\subseteq$ **Term**(τ).
- If $\alpha \in$ **Term**($\sigma \rightarrow \tau$) and $\beta \in$ **Term**(σ) then ($\alpha\,\beta$) $\in$ **Term**(τ) (function application).
- If $x \in$ **Var**(σ) and $\alpha \in$ **Term**(ρ), then $\lambda x.\alpha \in$ **Term**(τ), where $\tau = (\sigma \rightarrow \rho)$

Terms in Typed Lambda Calculus

We define terms **Term**(τ) of type τ:

- **Var**(τ) $\subseteq$ **Term**(τ).
- **Con**(τ) $\subseteq$ **Term**(τ).
- If $\alpha \in$ **Term**($\sigma \rightarrow \tau$) and $\beta \in$ **Term**(σ) then $(\alpha\,\beta) \in$ **Term**(τ) (function application).
- If $x \in$ **Var**(σ) and $\alpha \in$ **Term**(ρ), then $\lambda x.\alpha \in$ **Term**(τ), where $\tau = (\sigma \rightarrow \rho)$

Terms in Typed Lambda Calculus

We define terms **Term**(τ) of type τ:

- **Var**(τ) $\subseteq$ **Term**(τ).
- **Con**(τ) $\subseteq$ **Term**(τ).
- If $\alpha \in$ **Term**($\sigma \rightarrow \tau$) and $\beta \in$ **Term**(σ) then $(\alpha\,\beta) \in$ **Term**(τ) (function application).
- If $x \in$ **Var**(σ) and $\alpha \in$ **Term**(ρ), then $\lambda x.\alpha \in$ **Term**(τ), where $\tau = (\sigma \rightarrow \rho)$

Extending to a First Order Language

1. Variables i.e., **Var**(IND): $x, y, z, \ldots, x_0, x_1, x_2, \ldots$
2. Boolean connectives:

$\neg$	$\text{BOOL} \rightarrow \text{BOOL}$	(negation)
$\wedge$	$\text{BOOL} \rightarrow (\text{BOOL} \rightarrow \text{BOOL})$	(and)
$\vee$	$\text{BOOL} \rightarrow (\text{BOOL} \rightarrow \text{BOOL})$	(or)
$\rightarrow$	$\text{BOOL} \rightarrow (\text{BOOL} \rightarrow \text{BOOL})$	(if...then)

3. Quantifiers: $\forall$ (all), $\exists$ (some)
4. Equality:

$=$	$\tau \rightarrow (\tau \rightarrow \text{BOOL})$

5. Punctuation: brackets and period

Quantifier Syntax

- If $\phi \in$ **Term**(BOOL), and $x \in$ **Var**(IND), then $\forall x.\phi$ and $\exists x.\phi \in$ **Term**(BOOL).
- $x \in$ **Var**(IND) is called an individual variable.

Syntactic conventions:

- Instead of writing $((= \alpha)\beta)$, $((\wedge\phi)\psi)$, etc., we write $(= \alpha = \beta)$, $(\phi \wedge \psi)$, etc.
- Instead of writing e.g., ((chase fido) john), we sometimes write (chase fido john).
- **NB** this is equivalent to chase(john, fido) on a relational approach.

Quantifier Syntax

- If $\phi \in$ **Term**(BOOL), and $x \in$ **Var**(IND), then $\forall x.\phi$ and $\exists x.\phi \in$ **Term**(BOOL).
- $x \in$ **Var**(IND) is called an individual variable.

Syntactic conventions:

- Instead of writing $((= \alpha)\beta)$, $((\wedge\phi)\psi)$, etc., we write $(= \alpha = \beta)$, $(\phi \wedge \psi)$, etc.
- Instead of writing e.g., ((chase fido) john), we sometimes write (chase fido john).
- **NB** this is equivalent to chase(john, fido) on a relational approach.

Quantifier Syntax

- If $\phi \in$ **Term**(BOOL), and $x \in$ **Var**(IND), then $\forall x.\phi$ and $\exists x.\phi \in$ **Term**(BOOL).
- $x \in$ **Var**(IND) is called an individual variable.

Syntactic conventions:

- Instead of writing $((= \alpha)\beta)$, $((\wedge\phi)\psi)$, etc., we write $(= \alpha = \beta)$, $(\phi \wedge \psi)$, etc.
- Instead of writing e.g., ((chase fido) john), we sometimes write (chase fido john).
- **NB** this is equivalent to chase(john, fido) on a relational approach.

Quantifier Syntax

- If $\phi \in$ **Term**(BOOL), and $x \in$ **Var**(IND), then $\forall x.\phi$ and $\exists x.\phi \in$ **Term**(BOOL).
- $x \in$ **Var**(IND) is called an individual variable.

Syntactic conventions:

- Instead of writing $((= \alpha)\beta)$, $((\wedge\phi)\psi)$, etc., we write $(= \alpha = \beta)$, $(\phi \wedge \psi)$, etc.
- Instead of writing e.g., ((chase fido) john), we sometimes write (chase fido john).
- **NB** this is equivalent to chase(john, fido) on a relational approach.

Some Examples

1. $\exists x.(\text{love}\ x\ \text{kim})$
   Kim loves someone
2. $(\neg\exists x.(\text{love}\ x\ \text{kim}))$
   Kim doesn't love anyone
3. $\forall x.((\text{robber}\ x) \rightarrow \exists y.((\text{customer}\ y) \wedge (\text{love}\ y\ x)))$
   All robbers love a (perhaps different) customer
4. $\exists y.((\text{customer}\ y) \wedge \forall x.((\text{robber}\ x) \rightarrow (\text{love}\ y\ x)))$
   All robbers love the same customer

Some Examples

1. $\exists x.(\text{love}\ x\ \text{kim})$
   Kim loves someone
2. $(\neg\exists x.(\text{love}\ x\ \text{kim}))$
   Kim doesn't love anyone
3. $\forall x.((\text{robber}\ x) \rightarrow \exists y.((\text{customer}\ y) \wedge (\text{love}\ y\ x)))$
   All robbers love a (perhaps different) customer
4. $\exists y.((\text{customer}\ y) \wedge \forall x.((\text{robber}\ x) \rightarrow (\text{love}\ y\ x)))$
   All robbers love the same customer

Some Examples

1. $\exists x.(\text{love}\ x\ \text{kim})$
   Kim loves someone
2. $(\neg\exists x.(\text{love}\ x\ \text{kim}))$
   Kim doesn't love anyone
3. $\forall x.((\text{robber}\ x) \rightarrow \exists y.((\text{customer}\ y) \wedge (\text{love}\ y\ x)))$
   All robbers love a (perhaps different) customer
4. $\exists y.((\text{customer}\ y) \wedge \forall x.((\text{robber}\ x) \rightarrow (\text{love}\ y\ x)))$
   All robbers love the same customer

Some Examples

1. $\exists x.(\text{love}\ x\ \text{kim})$
   Kim loves someone
2. $(\neg\exists x.(\text{love}\ x\ \text{kim}))$
   Kim doesn't love anyone
3. $\forall x.((\text{robber}\ x) \rightarrow \exists y.((\text{customer}\ y) \wedge (\text{love}\ y\ x)))$
   All robbers love a (perhaps different) customer
4. $\exists y.((\text{customer}\ y) \wedge \forall x.((\text{robber}\ x) \rightarrow (\text{love}\ y\ x)))$
   All robbers love the same customer

Free and Bound Variables

$$((\text{customer}\ x) \vee \forall x.((\text{robber}\ x) \rightarrow \exists y.(\text{person}\ y)))$$

- First occurrence of x is free;
- Second occurrence of x is bound;
  Occurrence of y is bound.
- Free variable $\approx$ pronouns.
  - *She loves Fido*
- Context needed to interpret *she*;
  Something in addition to models so far needed to interpret free variables.

A WFF (**Term**(BOOL)) with no free variables is a (closed) sentence. FOL sentences $\subset$ WFFs.

Free and Bound Variables

$$((\text{customer}\ x) \vee \forall x.((\text{robber}\ x) \rightarrow \exists y.(\text{person}\ y)))$$

- First occurrence of x is free;
- Second occurrence of x is bound;
  Occurrence of y is bound.
- Free variable $\approx$ pronouns.
  - *She loves Fido*
- Context needed to interpret *she*;
  Something in addition to models so far needed to interpret free variables.

A WFF (**Term**(BOOL)) with no free variables is a (closed) sentence. FOL sentences $\subset$ WFFs.

Free and Bound Variables

$$((\text{customer}\ x) \vee \forall x.((\text{robber}\ x) \rightarrow \exists y.(\text{person}\ y)))$$

- First occurrence of x is free;
- Second occurrence of x is bound;
  Occurrence of y is bound.
- Free variable $\approx$ pronouns.
  - *She loves Fido*
- Context needed to interpret *she*;
  Something in addition to models so far needed to interpret free variables.

A WFF (**Term**(BOOL)) with no free variables is a (closed) sentence. FOL sentences $\subset$ WFFs.

Free and Bound Variables

$$((\text{customer}\ x) \vee \forall x.((\text{robber}\ x) \rightarrow \exists y.(\text{person}\ y)))$$

- First occurrence of x is free;
- Second occurrence of x is bound;
  Occurrence of y is bound.
- Free variable $\approx$ pronouns.
  - *She loves Fido*
- Context needed to interpret *she*;
  Something in addition to models so far needed to interpret free variables.

A WFF (**Term**(BOOL)) with no free variables is a (closed) sentence. FOL sentences $\subset$ WFFs.

Free and Bound Variables

$$((\text{customer}\ x) \vee \forall x.((\text{robber}\ x) \rightarrow \exists y.(\text{person}\ y)))$$

- First occurrence of x is free;
- Second occurrence of x is bound;
  Occurrence of y is bound.
- Free variable $\approx$ pronouns.
  - *She loves Fido*
- Context needed to interpret *she*;
  Something in addition to models so far needed to interpret free variables.

A WFF (**Term**(BOOL)) with no free variables is a (closed) sentence. FOL sentences $\subset$ WFFs.

Free and Bound Variables

$$((\text{customer}\ x) \vee \forall x.((\text{robber}\ x) \rightarrow \exists y.(\text{person}\ y)))$$

- First occurrence of x is free;
- Second occurrence of x is bound;
  Occurrence of y is bound.
- Free variable $\approx$ pronouns.
  - *She loves Fido*
- Context needed to interpret *she*;
  Something in addition to models so far needed to interpret free variables.

A WFF (**Term**(BOOL)) with no free variables is a (closed) sentence. FOL sentences $\subset$ WFFs.

Free and Bound Variables

$$((\text{customer}\ x) \vee \forall x.((\text{robber}\ x) \rightarrow \exists y.(\text{person}\ y)))$$

- First occurrence of x is free;
- Second occurrence of x is bound;
  Occurrence of y is bound.
- Free variable $\approx$ pronouns.
  - *She loves Fido*
- Context needed to interpret *she*;
  Something in addition to models so far needed to interpret free variables.

A WFF (**Term**(BOOL)) with no free variables is a (closed) sentence. FOL sentences $\subset$ WFFs.

Interpreting FOL Sentences

Task:

- Compute whether a sentence is *true* or *false* with respect to a model.
  - Is the sentence an accurate description of the situation?

Strategy: Compositionality!
Use recursion, but:

- Subformula of $\forall x.(\text{robber}\ x)$ is $(\text{robber}\ x)$ and this is not a sentence! So...

Satisfaction: $[\![\phi]\!]^{M,g} = 1$

Model M and variable assignment g satisfy the WFF ϕ.

- g defined for all individual variables, i.e., $x \in$ **Var**(IND);
- $g(x) \in D$.
- If α is an atomic term ($\in$ **Con**$(\tau) \cup$ **Var**(τ)), then

$$i_V^g(\alpha) = \begin{cases} g(\alpha) & \text{if } \alpha \text{ is a variable} \\ V(\alpha) & \text{if } \alpha \text{ is a constant} \end{cases}$$

- $[\![\exists x.\phi]\!]^{M,g} = 1$ iff $[\![\phi]\!]^{M,g[u/x]} = 1$ for some $u \in D$
- $g[u/x](x) = u$
- 'Try to find some value u for x that makes ϕ true'

Value of a term in a model

Where $M = \langle D, V \rangle$:

$[\![\alpha]\!]^{M,g} = i_V^g(\alpha)$	if	α is atomic
$[\![(\alpha\ \beta)]\!]^{M,g}$	$=$	$[\![\alpha]\!]^{M,g}([\![\beta]\!]^{M,g})$
$[\![(\lambda x.\alpha)]\!]^{M,g}$	$=$	that function h such that for any u $\in$ $D, h(u) = [\![\alpha]\!]^{M,g[u/x]}$
$[\![\alpha_1 = \alpha_2]\!]^{M,g} = 1$	iff	$[\![\alpha_1]\!]^{M,g} = [\![\alpha_2]\!]^{M,g}$
$[\![\neg\phi]\!]^{M,g} = 1$	iff	$[\![\phi]\!]^{M,g} = 0$
$[\![\phi \wedge \psi]\!]^{M,g} = 1$	iff	$[\![\phi]\!]^{M,g} = 1$ and $[\![\psi]\!]^{M,g} = 1$
$[\![\phi \vee \psi]\!]^{M,g}$	iff	$[\![\phi]\!]^{M,g} = 1$ or $[\![\psi]\!]^{M,g} = 1$
$[\![\phi \rightarrow \psi]\!]^{M,g} = 1$	iff	$[\![\phi]\!]^{M,g} = 0$ or $[\![\psi]\!]^{M,g} = 1$
$[\![\exists x.\phi]\!]^{M,g} = 1$	iff	$[\![\phi]\!]^{M,g[u/x]} = 1$ for some $u \in D$
$[\![\forall x.\phi]\!]^{M,g} = 1$	iff	$[\![\phi]\!]^{M,g[u/x]} = 1$ for every $u \in D$.

Truth (in terms of Satisfaction)

If $\phi \in$ **Term**(BOOL), we often write $M, g \models \phi$ instead of $[\![\phi]\!]^{M,g'} = 1$.

It doesn't matter which g you use for sentences, so:

Truth: A *sentence* ϕ is true in a model M (written $M \models \phi$) iff
for *any* g, $M, g \models \phi$

Validity: A sentence ϕ is valid (written $\models \phi$) iff
for any M, $M \models \phi$

Entailment: $\phi_1, \ldots, \phi_n \models \psi$ iff
if $M, g \models \phi_i$ for all i, $1 \leq i \leq n$, then $M, g \models \psi$