PHI 319: Philosophy, Computing, and Artificial Intelligence

Building Meaning With The Lambda Calculus

One way to generate the logical form of a sentence, and hence to associate truth-conditions with a sentence, is to use the lambda calculus to represent the semantics of the lexical items in the lexicon and to understand grammatical structure in terms of substitutions in the lambda calculus. (λ is the eleventh letter of the Greek alphabet.)

The Lambda Calculus

Here is a formula in the lambda calculus

λx.MAN(x)

In this formula, the prefix

λx

binds the occurrence of the variable

x

in the one-place predicate

MAN(x)

The Substitution Process

The formula **λx.MAN(x)** may be understood as a one-place function. The lambda binds a variable that is a placeholder for substitution of the argument in the one-place predicate. The symbol @ separates the functor from the argument. In the expression

λx.MAN(x)@vincent

The left hand expression

λx.MAN(x)

is the *functor*. The right hand expression

vincent

is the *argument*. The substitution process is called *β-reduction*. In the example, the result of the conversion is the sentence

MAN(vincent)

Grammatical Categories

This shows that we can think of the semantics of grammatical categories in terms of the lambda calculus. Here are the semantic representations of some lexical entries in the grammar

'every' : λP.λQ.∀x(P@x → Q@x)

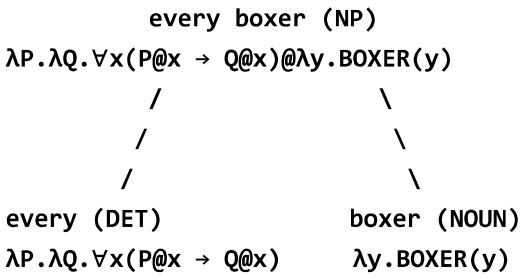
'some' or 'a' : λP.λQ.∃x(P@x ∧ Q@x)

'no' : λP.λQ.∀x(P@x → ¬Q@x)

'boxer' : λy.BOXER(y)

'walks' : λx.WALKS(x)

An example helps show how this works. Here is the representation tree for the noun phrase 'every boxer'



The expression at the root of the tree is

'every boxer'
λP.λQ.∀x(P@x → Q@x)@λy.BOXER(y)

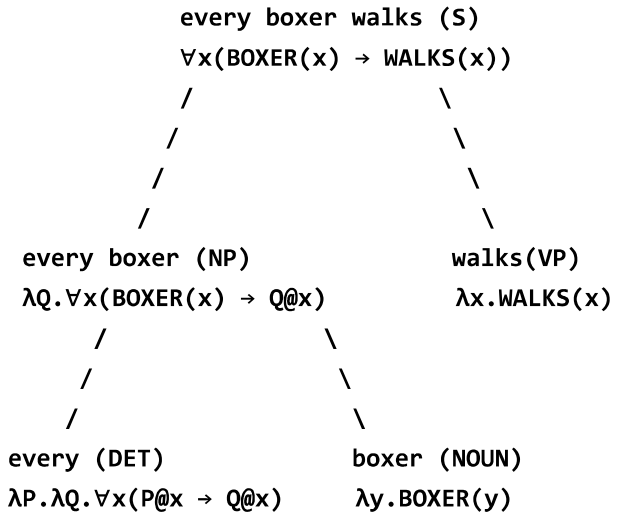
By substitution, this reduces to

'every boxer' (NP)
λQ.∀x(λy.BOXER(y)@x → Q@x)

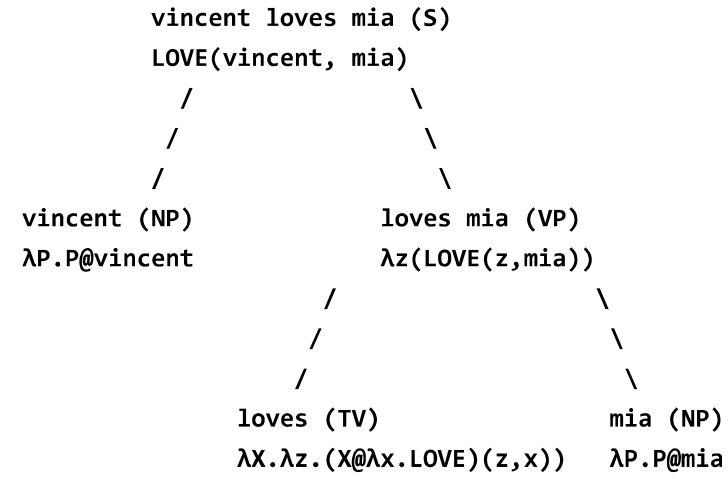
This reduces to

'every boxer' (NP)
λQ.∀x(BOXER(x) → Q@x)

If this noun phrase is given a verb phrase, the result is a sentence. Here, for example, is the representation tree (with substitutions completed) for the sentence 'every boxer walks'



Here is the representation tree for the sentence 'vincent loves mia'



Types

It is sometimes useful to think of lambda substitution in terms of types. There are two basic types, **e** and **t**. The first is the type of entities in the domain of the model. The second is the type for truth values (true and false). Compound types are built from these two basic types.

One place predicates, for example, have the type **<e, t>**. When a one-place predicate is given an expression of type **e**, it returns an expression with type **t**. To understand more clearly how this works, think again about the lambda expression

λx.MAN(x)@vincent

By β-conversion, this expression reduces to

MAN(vincent)

In terms of types, the lambda expression

```
λx.MAN(x)@vincent
<e, t>    e
```

reduces to

```
MAN(vincent)
t
```

A Simple Grammar to Illustrate Types and Type Conversion

Here is a simple grammar whose lexical entries are associated with formulas in the lambda calculus

S → VP NP
NP → NAME
NP → DET N
VP → IV
VP → TV NP

NAME → mia: λP.P@mia
NAME → vincent: λP.P@vincent

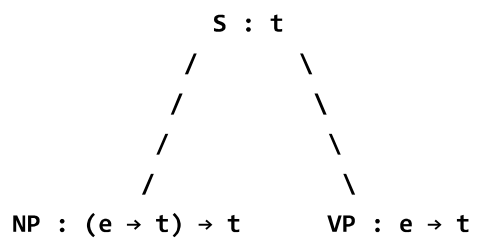
DET → every: λP.λQ.∀x(P@x → Q@x)
DET → some: λP.λQ.∃x(P@x ∧ Q@x)

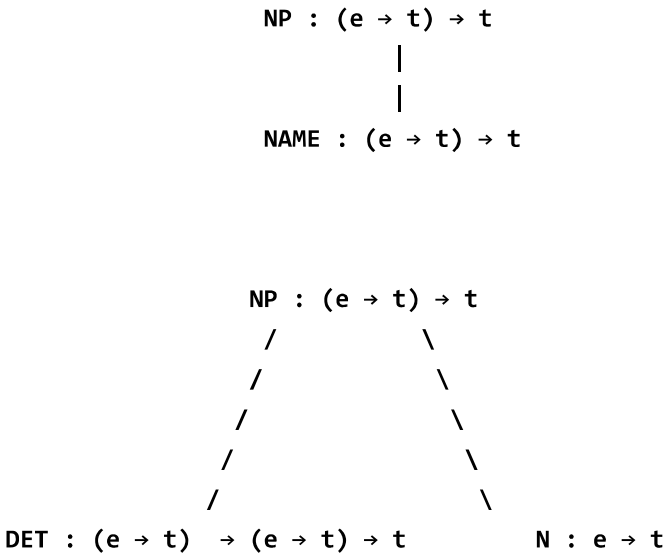
N → man: λx.MAN(x)
N → woman: λx.WOMAN(x)

IV → laughs: λx.LAUGHS(x)

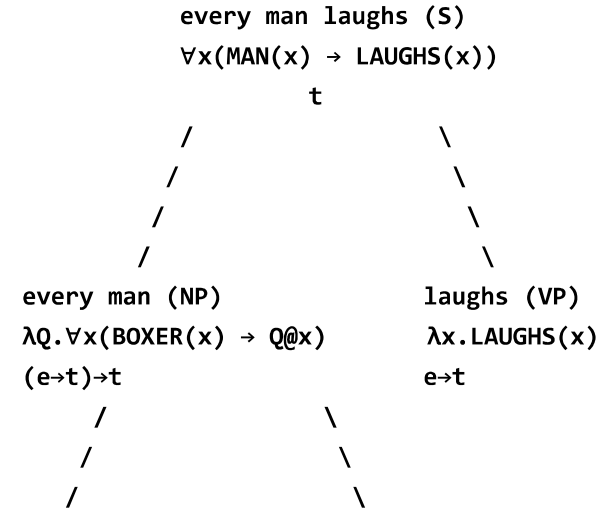
TV → loves: λX.λz.(X@λx.LOVE)(z,x))

Here are the abstract representation trees showing the type conversions





Here is the representation tree (with types and lambda formulas) for a specific grammatical sentence, 'every man laughs'



every (DET)	man (NOUN)
λP.λQ.∀x(P@x → Q@x)	λy.MAN(y)
(e →t)→(e→t)→t	e→t

Problems with Indefinite Noun Phrases in Discourse

This approach to generating the logical form of a sentence is powerful, but it is not without its problems. One such problem concerns the way indefinite NPs function in discourse.

In classical logic, $\exists x\varphi$ and $\neg\forall\neg x\varphi$ are truth conditionally equivalent. If φ is instantiated to

(dog(x) ∧ came_in(x))

then

∃x(dog(x) ∧ came_in(x))

and

¬∀x(dog(x) → ¬came_in(x))

are truth-functionally equivalent. But the sentences

A dog came in.

and

Not every dog failed to come in.

behave differently in discourse. The first can be extended in a conversation

A dog came in. It sat down.

The second cannot. The discourse

Not every dog failed to come in. It sat down.

does not make sense. This shows that truth conditions alone fail to capture the contextual dimension of sentence interpretation. The sentence

A dog came in.

updates the initially available context with an antecedent which can be picked up by anaphoric expressions in subsequent discourse. The truth conditionally equivalent

Not every dog failed to come in.

does not. This is a problem for the treatment of indefinite NPs (such as **a dog**) as **λP.λQ.∃x(P@x ∧ Q@x)**.

Problems with Indefinite Noun Phrases in Sentences

Indefinite NPs also present a problem in single sentences. It is traditional to illustrate this problem in terms of the so-called "donkey sentences."

(When Peter Geach first resented the phenomenon (now known as "donkey anaphora"), he used sentences about farmers and donkeys to construct his counterexample.
The word *anaphora* comes from the Greek word ἀναφορά, meaning "carrying back.")

The truth-conditions of the following two sentences

If John owns a donkey, he feeds it.

Every farmer who owns a donkey feeds it.

are

∀x((donkey(x) ∧ own(john, x)) → feeds(john,x))

∀x∀y((farmer(x) ∧ donkey(y) ∧ own(x, y)) → feeds(x, y))

In the first sentence, **a donkey** (located in the antecedent of a conditional) surfaces as a universally quantified expression with wide scope over the material conditional. In the second sentence, **a donkey** (located in relative clause that modifies a universally quantified NP) surfaces as a universally quantified expression taking wide scope. This contrasts with truth-conditions of sentences such as

A donkey came in.

where **a donkey** surfaces as an existentially quantified expression

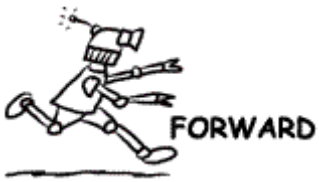
∃x(dog(x) ∧ came_in(x))

This too is a problem for the treatment of indefinite NPs (such as **a dog**) as **λP.λQ.∃x(P@x ∧ Q@x)**.

What is the solution?

One possibility is Discourse Representation Theory (DRT). This, however, is beyond the scope of this course.

Home



These lecture notes are incomplete. Comments or suggestions are welcome!

Last modified on 12/31/2016 21:18:20