An Inference Engine with a Natural Language Interface

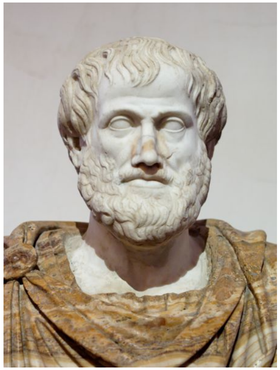
Jan van Eijck, CWI Amsterdam and Uil-OTS Utrecht

LOT Summer School, June 15, 2009

Overview

- The Aristotelian quantifiers
- A natural language engine for talking about classes.
- The example is taken from Jan van Eijck and Christina Unger, Computational Semantics with Functional Programming, Cambridge University Press 2009 (to appear).
- http://www.cwi.nl/~jve/cs/
- Demo
- A tentative connection with cognitive realities.

The Aristotelian quantifiers



All A are B	No A are B
Some A are B	Not all A are B

Aristotle interprets his quantifiers with existential import: All A are B and No A are B are taken to imply that there are A.

What can we ask or state with the Aristetelian quantifiers?

Questions and Statements (PN for plural nouns):

$$
\begin{array}{rcl}
Q & ::= & \text{Are all PN PN}? \\
 & | & \text{Are no PN PN}? \\
 & | & \text{Are any PN PN}? \\
 & | & \text{Are any PN not PN}? \\
 & | & \text{What about PN}?
\end{array}
$$

$$
\begin{array}{rcl}
S & ::= & \text{All PN are PN.} \\
 & | & \text{No PN are PN.} \\
 & | & \text{Some PN are PN.} \\
 & | & \text{Some PN are not PN.}
\end{array}
$$

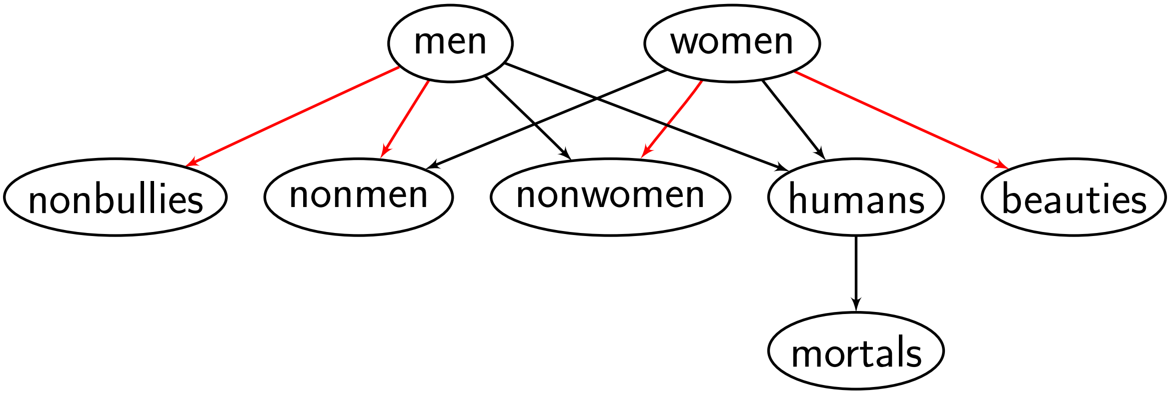
Example Interaction

```
jve@vuur:~/courses/lot2009$ ./Main
Welcome to the Knowledge Base.
Update or query the KB:
How about women?
All women are humans.
No women are men.

Update or query the KB:
All mammals are animals.
I knew that already.

Update or query the KB:
No mammals are birds.
```

```
OK.

Update or query the KB:
How about women?
All women are humans.
No women are men.

Update or query the KB:
All humans are mammals.
OK.

Update or query the KB:
How about women?
All women are animals.
All women are humans.
All women are mammals.
```

```
No women are birds.
No women are men.
No women are owls.

Update or query the KB:
```

Example Knowledge Base



The Meanings of the Aristotelean Quantifiers

What does ‘all’ mean? Inclusion.

What does ‘some’ mean? Non-empty intersection.

What does ‘not all’ mean? Non-inclusion.

What does ‘no’ mean? Empty intersection.

Key set-theoretic relation: inclusion

$$\subseteq$$

$A \subseteq B$ holds if and only if every element of A is element of B.

$A \not\subseteq B$ holds if and only if some element of A is not an element of B.

Complementation

Fix a universe U.

$\overline{A}$ denotes the set of things in the universe that are not elements of A.

$\overline{A}$ abbreviates $U - A$.

Building a Knowledge Base from Two Relations

The two relations we are going to model in the knowledge base are that of inclusion $\subseteq$ and that of non-inclusion $\not\subseteq$.

'all A are B' $\rightsquigarrow A \subseteq B$

'no A are B' $\rightsquigarrow A \subseteq \overline{B}$

'some A are not B' $\rightsquigarrow A \not\subseteq B$

'some A are B' $\rightsquigarrow A \not\subseteq \overline{B}$ (equivalently: $A \cap B \neq \emptyset$).

Knowledge Base: Definition

The two **booleans** are $\top$ (true) and $\bot$ (false).

A **knowledge base** is a list of triples

$$(\text{Class}_1, \text{Class}_2, \text{Boolean})$$

where $(A, B, \top)$ expresses that $A \subseteq B$,

and $(A, B, \bot)$ expresses that $A \not\subseteq B$.

Rules of the Inference Engine

Let $\widetilde{A}$ be given by: if A is of the form $\overline{C}$ then $\widetilde{A} = C$, otherwise $\widetilde{A} = \overline{A}$. Let $A \Longrightarrow B$ express $A \subseteq B$. Let $A \not\Longrightarrow B$ express $A \not\subseteq B$.

Computing the subset relation from the knowledge base:

$$\frac{(A, B, \top) \in \mathbf{K}}{A \Longrightarrow B} \qquad \frac{A \Longrightarrow B}{\widetilde{B} \Longrightarrow \widetilde{A}} \qquad \frac{A \Longrightarrow B \qquad B \Longrightarrow C}{A \Longrightarrow C}$$

Computing the non-subset relation from the knowledge base:

$$\frac{(A, B, \bot) \in \mathbf{K}}{A \not\Longrightarrow B} \qquad \frac{A \not\Longrightarrow B}{\widetilde{B} \not\Longrightarrow \widetilde{A}} \qquad \frac{A \Longleftarrow B \quad B \not\Longrightarrow C \quad C \Longleftarrow D}{A \not\Longrightarrow D}$$

Reflexivity and existential import:

$$\frac{}{A \Longrightarrow A} \qquad \frac{}{A \not\Longrightarrow \widetilde{A}}\ A \text{ not of the form } \overline{C}$$

Consistency of a Knowledge Base

A Knowledge Base **K** is inconsistent if for some $A \Longrightarrow B$:

$$\frac{\mathbf{K}}{A \Longrightarrow B} \qquad \frac{\mathbf{K}}{A \not\Longrightarrow B}$$

Otherwise **K** is consistent.

Soundness and Completeness of Inference System

Exercise 1 *An inference system is called sound if all conclusions that can be derived are valid, i.e. if all axioms are true and all inference rules preserve truth. Show that the inference system for Aristotelian syllogistics is sound.*

Exercise 2 *An inference system is called complete if it can derive all valid conclusions from a set of premisses. In other words: if* $A \Longrightarrow B$ *does not follow from a knowledge base, then there is a class model for the knowledge base where* $A \not\subseteq B$*, and if* $A \not\Longrightarrow B$ *does not follow from a knowledge base, then there is a class model for the knowledge base where* $A \subseteq B$*. Show that the inference system for Aristotelian syllogistics is complete.*

Implementation (in Haskell)

In our Haskell implementation we can use `[(a,a)]` for relations.

```
type Rel a = [(a,a)]
```

The composition of two relations R and S on A us the set of pairs

$$\{(x, y) \mid \exists z \in A : (x, z) \in R \text{ and } (z, y) \in S\}$$

```
(@@) :: Eq a => Rel a -> Rel a -> Rel a
r @@ s = nub [ (x,z) | (x,y) <- r, (w,z) <- s, y == w ]
```

`Eq a` indicates that a is in the equality class.

Least Fixpoint Computation and Transitive Closure

Least Fixpoint Computation: gradually getting there ...

```
lfp :: Eq a => (a -> a) -> a -> a
lfp f x | x == f x  = x
        | otherwise = lfp f (f x)
```

Transitive closure of a relation R: least transitive relation that includes R.

Computation of transitive closure by 'making the relation transitive' in stages.

This uses the operation for least fixpoint: TC$(R) =$ lfp$(\lambda S.S \cup R \cdot S)R$.

```
tc :: Ord a => Rel a -> Rel a
tc r = lfp (\ s -> (sort.nub) (s ++ (r@@s))) r
```

Least Fixpoint Computation and Reflexive Transitive Closure

Reflexive transitive closure of a relation R: least reflexive and transitive relation that includes R.

```
rtc :: Ord a => [a] -> Rel a -> Rel a
rtc xs r = lfp (\ s -> (sort.nub) (s++(r@@s))) i
           where i = [ (x,x) | x <- xs ]
```

This uses:

$\mathsf{RTC}(R) = \mathsf{lfp}(\lambda S.S \cup R \cdot S)I$.

Classes and Opposite Classes

Assume that each class has an opposite class. The opposite of an opposite class is the class itself.

```
data Class = Class String | OppClass String
             deriving (Eq,Ord)

instance Show Class where
  show (Class xs)    = xs
  show (OppClass xs) = "non-" ++ xs

opp :: Class -> Class
opp (Class name)    = OppClass name
opp (OppClass name) = Class name
```

Declaration of the knowledge base

```
type KB = [(Class, Class, Bool)]
```

A data type for statements and queries:

```
data Statement =
     All  Class   Class | No      Class Class
   | Some Class   Class | SomeNot Class Class
   | AreAll Class Class | AreNo   Class Class
   | AreAny Class Class | AnyNot  Class Class
   | What   Class
  deriving Eq
```

Negations of queries

```
neg :: Statement -> Statement
neg (AreAll as bs) = AnyNot as bs
neg (AreNo as bs)  = AreAny as bs
neg (AreAny as bs) = AreNo as bs
neg (AnyNot as bs) = AreAll as bs
```

Use the reflexive transitive closure operation to compute the subset relation from the knowledge base.

```
subsetRel :: KB -> [(Class,Class)]
subsetRel kb = rtc
  (domain kb) ([(x,y)           | (x,y,True) <- kb ]
           ++  [(opp y,opp x) | (x,y,True) <- kb ])
```

This uses the following function for getting the domain of a knowledge base:

```
domain :: [(Class,Class,Bool)] -> [Class]
domain = nub . dom where
  dom [] = []
  dom ((xs,ys,_):facts) =
             xs : opp xs : ys : opp ys : dom facts
```

Supersets of a class

If $R \subseteq A^2$ and $x \in A$, then $xR := \{y \mid (x, y) \in R\}$.

```
rSection :: Eq a => a -> Rel a -> [a]
rSection x r = [ y | (z,y) <- r, x == z ]
```

The supersets of a class are given by a right section of the subset relation. I.e. the supersets of a class are all classes of which it is a subset.

```
supersets :: Class -> KB -> [Class]
supersets cl kb = rSection cl (subsetRel kb)
```

Similarly, compute the non-subset relation from the knowledge base (see Section 5.7 in book draft for details).

The non-supersets of a class:

```
nsupersets :: Class -> KB -> [Class]
nsupersets cl kb = rSection cl (nsubsetRel kb)
```

Query of a knowledge base

By means of yes/no questions:

```
deriv :: KB -> Statement -> Bool
deriv kb (AreAll as bs) = elem bs (supersets as kb)
deriv kb (AreNo  as bs) = elem (opp bs) (supersets as kb)
deriv kb (AreAny as bs) = elem (opp bs) (nsupersets as kb)
deriv kb (AnyNot as bs) = elem bs (nsupersets as kb)
```

Caution

There are three possibilities:

- `deriv kb stmt` is true. This means that the statement is derivable, hence true.
- `deriv kb (neg stmt)` is true. This means that the negation of `stmt` is derivable, hence true. So `stmt` is false.
- neither `deriv kb stmt` nor `deriv kb (neg stmt)` is true. This means that the knowledge base has no information about `stmt`.

Building a KB

To build a knowledge base we need a function for updating an existing knowledge base with a statement.

If the update is successful, we want an updated knowledge base. If it is not, we want to get an indication of failure.

Example: Update with an 'All' statement

The update function checks for possible inconsistencies. E.g., a request to add an $A \subseteq B$ fact to the knowledge base leads to an inconsistency if $A \not\subseteq B$ is already derivable.

```
update  :: Statement -> KB -> Maybe (KB,Bool)

update (All as bs) kb
  | elem bs (nsupersets as kb)  = Nothing
  | elem bs (supersets as kb)   = Just (kb,False)
  | otherwise                   =
                  Just (((as,bs,True): kb),True)
```

```
mytxt = "all bears are mammals\n"
     ++ "no owls are mammals\n"
     ++ "some bears are stupids\n"
     ++ "all men are humans\n"
     ++ "no men are women\n"
     ++ "all women are humans\n"
     ++ "all humans are mammals\n"
     ++ "some men are stupids\n"
     ++ "some men are not stupids"
```

```
Main> process mytxt
[(men,stupids,False),(men,non-stupids,False),
 (humans,mammals,True),(women,humans,True),
 (men,non-women,True),(men,humans,True),
 (bears,non-stupids,False),(owls,non-mammals,True),
 (bears,mammals,True)]
```

Demo

. . .

Conclusions

- Mini-case of computational semantics. What is the use of this?
- Cognitive research focusses on this kind of quantifier reasoning . . .
- Can this be used to meet cognitive realities? Links with cognition by refinement of this calculus . . . The "natural logic for natural language" enterprise: special workshop during Amsterdam Colloquium 2009 (see `http://www.illc.uva.nl/AC2009/`
- Towards Rational Reconstruction of Cognitive Processing
- Tomorrow: more about sets, functions and types.
- Lots of interconnections with Yoad Winter's course.

References

[1] Jan van Eijck. DEMO — a demo of epistemic modelling. In Johan van Benthem, Dov Gabbay, and Benedikt Löwe, editors, Interactive Logic — Proceedings of the 7th Augustus de Morgan Workshop, number 1 in Texts in Logic and Games, pages 305–363. Amsterdam University Press, 2007.

[2] R. Frost and J. Launchbury. Constructing natural language interpreters in a lazy functional language. The Computer Journal, 32(2):108–121, 1989.

[3] Richard A. Frost. Realization of natural language interfaces using lazy functional programming. ACM Comput. Surv., 38(4), 2006.

[4] R. Montague. The proper treatment of quantification in ordinary English. In J. Hintikka, editor, Approaches to Natural Language, pages 221–242. Reidel, 1973.

[5] R. Montague. English as a formal language. In R.H. Thomason, editor, Formal Philosophy; Selected Papers of Richard Montague, pages 188–221. Yale University Press, New Haven and London, 1974.

[6] R. Montague. Universal grammar. In R.H. Thomason, editor, Formal Philosophy; Selected Papers of Richard Montague, pages 222–246. Yale University Press, New Haven and London, 1974.

[7] F.C.N. Pereira and H.D. Warren. Parsing as deduction. In Proceedings of the 21st Annual Meeting of the ACL, pages 137–111. MIT, Cambridge, Mass., 1983.

[8] S.M. Shieber. An Introduction to Unification Based Approaches to Grammar, volume 4 of CSLI Lecture Notes. CSLI, Stanford, 1986. Distributed by University of Chicago Press.