Lecture 15: Quantifiers & Typed Logic

CS 181O
Spring 2016
Kim Bruce

Interpreting language

- Two options: indirect & direct

Natural language expression

Logical representation

Model-theoretic interpretation

Done!

Conditions on Quantifiers

- Write $D_E AB$ to stand for determiner expression (like those on previous slide) with E the domain of discourse, A the restriction and B its body.
  - E.g., "Every dog barked" has dog(x) as restriction and barked(x) as the body.
  - Similarly for "A dog barked" or "Most dogs barked"

Conditions on Quantifiers

- Require:
  - EXT: For all A, B $\subseteq$ E $\subseteq$ E', $D_E AB \Leftrightarrow D_{E'} AB$
    - *Extension*
    - Expanding the domain makes no difference to truth if A, B fixed.
    - Really, only $A \cup B$ matters
  - CONS: For all A, B $\subseteq$ E $\subseteq$ E', $D_E AB \Leftrightarrow D_{E'} A(A \cap B)$
    - *Conservativity*
    - For the body, only the elements in the body matter
    - Not hold of "Only dogs barked"
  - EXT + CONS $\Rightarrow$ Only A-B and $A \cap B$ matter in determining truth of $D_E AB$

Expressing Quantifiers

- Quantifiers can be expressed using only $|A \cap B|$ and $|A - B|$
  - All A are B $\Rightarrow |A - B| = 0$
  - Some A are B $\Rightarrow |A \cap B| > 0$
  - Most A are B $\Rightarrow |A \cap B| > |A - B|$

Further Conditions

- For quantifiers on quantity:
  - ISOM: If f is a bijection from E to E',
    then $D_E AB \Leftrightarrow D_{E'} f[A]\ f[B]$
- A quantifier is a relation Q satisfying EXT, CONS, and ISOM
- Characterize according to $|A-B|$ and $|A \cap B|$

Tree of Numbers

- Record pairs corresponding to $|A-B|$ and $|A \cap B|$
- Structure:

$|A| = 0$ 0,0

$|A| = 1$ 1,0 0,1

$|A| = 2$ 2,0 1,1 0,2

$|A| = 3$ 3,0 2,1 1,2 0,3

$|A| = 4$ 4,0 3,1 2,2 1,3 0,4

Tree of Numbers

- At least two:

−

− −

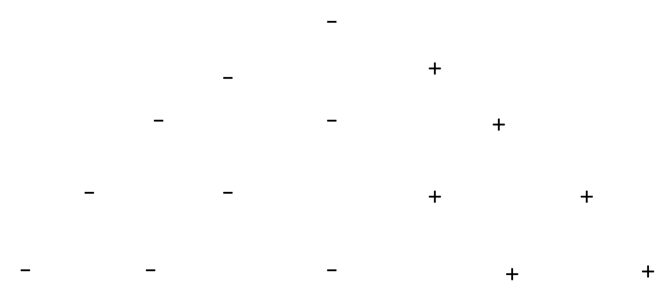
− − +

− − + +

− − + + +

Tree of Numbers

- Most:



Alternative representation

- λ-calculus in terms of m = |A - B|, n = |A∩B|
  - At least two: λm λn. n ≥ 2
  - Most: λm λn. n > m
  - No: λm λn. n = 0

More Properties

- Reflexive: ∀X. QXX
  - holds of "all" and "exists" but not "no" or "not all"
- Symmetric: ∀X∀Y. (Q X Y ⇔ Q Y X)
  - holds of "exists" and "no", but not "all" or "not all"
- Upward right monotonic: Q A B and B ⊆ B' implies Q A B'
  - holds of all, exists, at least n, but not "no"
- Downward right monotonic: Q A B and B' ⊆ B implies Q A B'
  - holds of "not all" and "no"

Typed Logic

- Text shifts to typed logic (really typed lambda calculus) to help move directly to interpret natural language in a model.
  - Keep track of types of variables, constants, functions, and relations.
  - type ::= e | t | (type → type)
  - exp ::= c | vble | λv:type . exp | (exp exp)
- But expressions must be well typed!

Model

- Model M:
  - Start with domain D_e, and then build up other domains using →
  - Comes with interpretation function I for constants, functions, and relations — interprets in the appropriate types.

Model

- Defining meaning in M with typed variable assignment g:
  - $[[c]]_g^M = I(c)$
  - $[[x]]_g^M = g(x)$
  - $[[\lambda v : \tau . E]]_g^M = h$
    where $h : D_\tau \to D_\tau$ is the function defined by $\lambda d : D_\tau . [[E]]_{g[v:=d]}^M$

    $[[(E_1 E_2)]]_g^M = [[E_1]]_g^M([[(E_2)]]_g^M)$

What about logic?

- Logical operators just treated as constants in lambda calculus:
  - [[¬]] = h where h = λp. not p
  - [[∧]] = h where h = λp. λq. p && q
  - [[∨]] = h where h = λp. λq. p || q

∀ and ∃ are a bit trickier as they bind variables

Quantifiers

- Treat quantifiers as operators on functions:
  - ∀x.E encoded as ∀(λx.E) and ∃x.E encoded as ∃(λx.E)
  - [[∀]] = h where h: (e → t) → t is defined s.t. for f: e → t
    h(f) = True iff f(d) = True for all d in e, and = False otherwise
  - [[∃]] = h where h: (e → t) → t is defined s.t. for f: e → t
    h(f) = False iff f(d) = False for all d in e, and = True otherwise
  - Could add quantifiers for higher types, but won't bother for now.

Predicate Logic in Typed Logic

- All formulas encoded, e.g.
  - $P\ x \wedge Q\ y \equiv \wedge\ (P\ x)\ (Q\ y)$
  - $\forall x.\ P\ x \equiv \forall\ (\lambda x.\ P\ x)$
  - Usually write in infix anyway
- No longer have to worry about separate translation to predicate calculus and then interpret in model.

Computing Truth

- Use α-conversion, and β & η-reduction as before to compute values
- Define substitution (written $E[x := s]$) as before (see text).
- Note typos in book on definitions of α conversion and η-reduction.

Nice Properties

- Confluence: If M can be reduced to a normal form, then there is only one such normal form.
- Normal Form: Every expression of typed logic can be reduced to a normal form *(not true of untyped lambda calculus)*

Semantics

- Described in file TCOM.hs
- Returns value in given model (Model.hs)
- Most cases similar to before though return Bools rather than logical formulas.
- Big differences in determiners

Questions?