Lecture notes by Edward Loper

Course: Ling 554 (Type-Logical Semantics)
Professor: Bob Carpenter
Institution: University of Pennsylvania

Tuesday, October 3, 2000

1 Review of update semantics

- distinction between world knowledge & discoures logic.
- divide world into referential part and propostional part..
- nonrigid designation
- extension of scope of $\exists$, not $\forall$.
- We can get de re (belief about object) vs de dicto (belief about description) distinctions.

2 λ calculus and type theory

Define 2 $\perp$ types: e (entity), t (truth value).

Walks: e$\rightarrow$ t

Define: BasTyp = {Ind, Bool}

interpret concat as modus ponens or functional application.

2.1 Toy Language

1. var_τ: a countably infinite set of type τ
2. con_τ: a set of constants of type τ
3. Var = $\cup_($$\tau \in$ Typ) Var_τ
4. Con = $\cup_($$\tau \in$ Typ) Con_τ

terms:

1. $\text{var}_\tau \subset \text{Term}_\tau$
2. $\text{con}_\tau \subset \text{Term}_\tau$
3. function application
4. lambda abstraction: λ x.(a) yields the appropriate type.

Free variables vs. bound variables..

Substitution: $\alpha[\text{x}\mapsto\beta]$

FreeFor(α,x,β): is α free for x in β?

A model is: M = $\langle$ Dom, $[\![\bullet]\!]\rangle$

We still need the equiv of our g function: θ: Var $\rightarrow$ Dom, s.t. $\theta(\text{x}) \in \text{Dom}_\tau$ if x $\in \text{Var}_\tau$

denotations: $[\![\alpha]\!]_{\text{M}}{}^\theta$

2.2 Properties

- system is sound: if α is type τ, $[\![\alpha]\!] \in \text{Dom}_\tau$, for every θ and M.
- bound variables' names unimportant
- logical equivlanance if denotations are equal..

Type of $\wedge$ is bool$\rightarrow$ bool$\rightarrow$ bool.

order in which a function recieves its arguments is arbitrary.

consider: John loves and Mary hates apple pie.

give "John loves" an interpretation by permuting the lambda variables.

Define composition: $(\beta \circ \alpha)(\delta) = \beta(\alpha(\delta))$

lets us do things like combining "carefully walk" before applying it..

α reduction = substitute a bound variable β reduction = apply a function η reduction = λ x(α(x)) $\mapsto \alpha$ if x not in free(α)

Other properties:

- reflexivity
- transitivity
- congruance: $\alpha \mapsto \alpha'$, $\beta \mapsto \beta' \vdash \alpha(\beta) \mapsto \alpha'(\beta')$
- congruance on lambda abstraction..
- equivalance

reductions are confluent (church-rosser) reduction eventually halts for any finite expression

we can define notion of proof.

Define normal forms.. β normal form means there are no more β reductions that you can do, etc.

If α and β are in normal form, $\alpha \equiv \beta$ iff $\alpha =_\alpha \beta$

completeness: two λ-terms α and β are logically equivalant only if $\vdash \alpha \Leftrightarrow \beta$ is provable.

decidability: there is an algorithm for deciding whether 2 terms are logically equivlant.

Tuesday, October 10, 2000

Single functor/single term. But do we only have binary branching? Functions might take multiple args..

So define product times: $(\sigma \times \tau) \in$ Typ if $\sigma, \tau \in$ Typ

$[\![$ Give$]\!]([\![$ John$]\!], [\![$ Book$]\!])$

Define new constants and variables of product type. Does NL have product type constants?

Need prrojection functions:

- $\pi_1(\alpha)$ gives 1st element
- $\pi_2(\alpha)$ gives 2nd element

$\text{Dom}_{\sigma \times \tau} = \text{Dom}_\sigma \times \text{Dom}_\tau$

Define operators on terms.. curry/uncurry, commute and reassociate.

2.3 Applicative Categorial Grammar

Start with a basic set of categories, BasCat (np, n, s).

Define them as:

- np: ind
- n: ind->bool
- s: bool

Define Cat:

1. BasCat $\subseteq$ Cat
2. If A, B $\in$ Cat then (A/B), (A{\}B) $\in$ Cat

- A/B is the forward functor with domain (arg) B and range (result) A.
- B{\}A is the backward functor with domain (arg) B and range (result) A.

(B B{\}A) $\rightarrow$ A (A/B B) $\rightarrow$ A

Typ(A/B) = Typ(B{\}A) = Typ(B) $\rightarrow$ Typ(A)

VP: Typ(np{\}s) = Typ(np) $\rightarrow$ Typ(s) = ind $\rightarrow$ bool

abbreviate lexical entries as: e $\Rightarrow$ α: A = <e, <A, a>>

<kiss, <((np{\}s)/np), (ind$\rightarrow$(ind$\rightarrow$ bool))>

np{\}s: expects an np on the left, gives an s. np{\}s/np: expects an np on left and right, gives an S. np{\}s/np/np: expects 1 np on left, 2 on right, gives s.

Proof tree:

```
 Bobb Barr          sneezes
----------- Lx     ------------ Lx
Bobbie: np
```

3 Game theoretical semantics

Hintika: the principles of mathmatics revisited

We are given a first-order language L and a model M of L.

Define a two-person game G(S; M)

1. Two players:
   - myself: the initial verifier
   - nature: the initial falsifier
   - At each stage of the game, the verifier is trying to show S is

   true in M, and the falsifier that it's false.
2. Everything gets named

A sentence is true if the verifier has a winning strategy. A sentence is false if the falsifier has a winning strategy.

Theorem: for any 1st-order sentene, tarski-type truth and GTS truth coincide.

A sigma(1,1) sentence is a second order existential sentence. e.g., (∃ f1, f2)(∀ x)[[f2(x)=0∧ R(...)]]

In ∀ x∃ yRxy, choice of y depends on x.

Introduce: (∃ y/∀ x) means the choice of y is independant of x.

Consider: some representative from every village met some relative of every townsman.

3.1 Partiality

Assign expressions one of 3 values: 0, 1, and ?. Use positive and negative extensions of predicates:

1. P(A) = 1 if a ∈ P+
2. P(A) = 0 if a ∈ P-
3. P(A) = ? if (a ∉ P+) and (a ∉ P-)

Strong Kleene: (1∨? = 1) Bochvar: (1∨? = ?)

it's important to prove that we'll never get a sentence that's both true and false..

3.2 Consequences

IF logic is not compositional in the ordinary sense! When we get down to (∃ x/∀ y)S[x], we need to know about y... We can't just use substitution..

3.3 Epistemic Logic

Define Ka as an operator, intuitively interpreted as "a knows that ...".

Each world M_0 ∈ W and each person b existing in M_0 is associated with a set of worlds, the epistemic b-alternatives to M_0.

Let Ω be a model structure and M_0∈Ω.. Then Ka(S) in M_0 iff for each epistemic a-alternative M_1 to M_0 in Ω, S is true...

(R.K) The game G(Ka(S); M_0) begins with a choice by the falsifier of an epistemic a-alternative M_1 to M_0. Continue as G(S; M_1)

3.4 Natural Language

Assert that there are no overt quantifier-variable pairings.. Modify game rules so names for individuals are substituted for entire generalized quantifiers (= Det N).

Treat interpretation of sentences as subgames. Individuals used for a subgame G(S;M) ust be selected from a choice set Is..

4 Sequent Calculus

Treat proof rules as arrays: record the entailment relations as you go along. Each node records a set of premises and a conclusion.

You can treat Γ as a finite conjunction of formulas.

4.1 Semantic Tableaux

"branches close" $\rightarrow$ inconsistant

$\Gamma \models \varphi$

Either show that a branch closes (inconsistancy) or no branch closes.

Use rules to keep rewriting the set, until we get to the end.

Contradiction:

$\Gamma \models \bot$

Consistant:

$\Gamma \models$ `something`
$\Gamma \neg\models \bot$

Rules:

```
Γ, φ ∧ ψ consistant
---------------------
Γ, φ ∧ ψ, φ, ψ consistant
```

```
Γ, ¬(φ ∧ ψ) consistant
---------------------
Γ, ¬(φ ∧ ψ), ¬ ψ consistant OR Γ, ¬(φ ∧ ψ), ¬φ, consistant
```

At any point, we're keeping track of a set of possible consistant assertions.

Simplify by eliminating repeating conjunctions:

```
Γ, φ ∧ ψ consistant
---------------------
Γ, φ, ψ consistant
```

Turn it up side down and invert consistant:

```
Γ, φ ∧ ψ, φ, ψ ⊨⊥
---------------------
Γ, φ ∧ ψ   ⊨⊥
```

We can write $\models\bot$ as $\Rightarrow$ with nothing on the right

4.2 Rules

Closing:

```
----------- (basic segment)
Γ, φ, ¬φ ⇒
```

```
Γ, φ, ψ ⇒
--------------
Γ, φ∧ψ ⇒
```

```
Γ, φ, ψ ⇒
--------------
Γ, φ∧ψ ⇒
```

```
Γ,φ ⇒    Γ,ψ ⇒
----------------
  Γ, φ∨ψ ⇒
```

```
Γ,¬φ ⇒     Γ,ψ ⇒
-----------------
   Γ, φ→ψ ⇒
```

```
Γ, φ, ¬ψ
--------------
Γ, ¬(φ→ψ) ⇒
```

```
Γ, φ ⇒
---------
Γ, ¬ ¬ φ ⇒
```

For Tableaux:

```
     Γ, ∀ xφ ⇒
------------------
Γ, ∀ xφ, φ(x/x) ⇒
```

The following are equivlanat:

```
 Γ ⇒ φ
---------
Γ, ¬φ ⇒
```

Use that to simplify to things like:

```
Γ ⇒ φ      Γ ⇒ ψ
-------------------
     Γ ⇒ φ∧ψ
```

In a linguistics domain, rules will be things like:

```
NP VP ⇒ S
```

Which means that basically we have a CFG here (it's equivalant)..

4.3 Gentzen Sequents

Allow sequents to have any finite number of formulas on both the left AND the right side:

```
Γ ⇒ Δ
```

Means that if all formulas in Γ are true, then at least one formula in Δ is true.

see slides p. 12

Now, sequents are no longer equivalant to rewrite rules, since there can be more than one thing on the right..