Lecture 12: Language to Logic

CS 181O
Spring 2016
Kim Bruce

Some slide content taken from Unger and Michaelis

Last Time ...

- Let e be type of elements in universe, t be truth values.
- Type of sentence is t
- Meaning and types of determiners
  - "a" $\Rightarrow \lambda Q: e \rightarrow t.\ \lambda P: e \rightarrow t.\ \exists x: e.\ (Q(x) \wedge P(x))$
  - "every" $\Rightarrow \lambda Q: e \rightarrow t.\ \lambda P: e \rightarrow t.\ \forall x: e.\ (Q(x) \rightarrow P(x))$
  - "the" $\Rightarrow \lambda P: e \rightarrow t.\ \lambda Q: e \rightarrow t.$
    $\exists x: e.\ \forall y: e.\ ((P(y) \leftrightarrow x = y) \wedge Q(x))$
- All have type $(e \rightarrow t) \rightarrow (e \rightarrow t) \rightarrow t$

More types:

- Type of determiner: $(e \rightarrow t) \rightarrow (e \rightarrow t) \rightarrow t$
- Type of Noun?
  - $e \rightarrow t$
- Type of noun phrase?
  - $(e \rightarrow t) \rightarrow t$
- Type of verb phrase?
  - $e \rightarrow t$
- Type of sentence? t!

Natural Language Semantics

- Take grammar from lecture 6 and translate sentences to predicate logic.
- Use lambda calculus for semantics of phrases.
- Compose using function application.
- Meaning of sentence is formula of predicate logic.

Grammar from lecture 6

- S → NP VP
- NP → Snow White | Alice | Dorothy | Goldilocks | DET CN | DET RCN
- DET → the | every | some | no
- CN → girl | boy | princess | dwarf | giant | sword | dagger
- RCN → CN that VP | CN that NP TV
- VP → laughed | cheered | shuddered | TV NP | DV NP NP
- TV → loved | admired | helped | defeated | caught
- DV → gave

Syntax

- Review FSynF.hs

```
data Term = Var Variable | Struct String [Term]
        deriving (Eq,Ord)
data Formula a = Atom String [a]
              | Eq a a
              | Neg  (Formula a)
              | Impl (Formula a) (Formula a)
              | Equi (Formula a) (Formula a)
              | Conj [Formula a]
              | Disj [Formula a]
              | Forall Variable (Formula a)
              | Exists Variable (Formula a)
              deriving Eq
```

Translating

```
type LF = Formula Term — LF is logical formula

lfSent :: Sent -> LF
lfSent (Sent np vp) = (lfNP np) (lfVP vp)

lfNP :: NP -> (Term -> LF) -> LF
lfNP SnowWhite     = \ p -> p (Struct "SnowWhite"  [])
lfNP Alice         = \ p -> p (Struct "Alice"      [])
lfNP Dorothy       = \ p -> p (Struct "Dorothy"    [])
lfNP Goldilocks    = \ p -> p (Struct "Goldilocks" [])
lfNP LittleMook    = \ p -> p (Struct "LittleMook" [])
lfNP Atreyu        = \ p -> p (Struct "Atreyu"     [])
lfNP (NP1 det cn)  = (lfDET det) (lfCN cn)
lfNP (NP2 det rcn) = (lfDET det) (lfRCN rcn)
```

From MCWPL.hs

Translating

```
lfVP :: VP -> Term -> LF
lfVP Laughed   = \ t -> Atom "laugh"   [t]
lfVP Cheered   = \ t -> Atom "cheer"   [t]
lfVP Shuddered = \ t -> Atom "shudder" [t]

lfVP (VP1 tv np) =
   \ subj -> lfNP np (\ obj -> lfTV tv (subj,obj))
lfVP (VP2 dv np1 np2) =
   \ subj -> lfNP np1 (\ iobj -> lfNP np2 (\ dobj ->
                  lfDV dv (subj,iobj,dobj)))

lfTV :: TV -> (Term,Term) -> LF
lfTV Loved    = \ (t1,t2) -> Atom "love"   [t1,t2]
lfTV Admired  = \ (t1,t2) -> Atom "admire" [t1,t2]
lfTV Helped   = \ (t1,t2) -> Atom "help"   [t1,t2]
lfTV Defeated = \ (t1,t2) -> Atom "defeat" [t1,t2]
```

Translating

```
lfDV :: DV -> (Term,Term,Term) -> LF
lfDV Gave = \ (t1,t2,t3) -> Atom "give" [t1,t2,t3]

lfCN :: CN -> Term -> LF
lfCN Girl     = \ t -> Atom "girl"     [t]
lfCN Boy      = \ t -> Atom "boy"      [t]

lfCN Princess = \ t -> Atom "princess" [t]
lfCN Dwarf    = \ t -> Atom "dwarf"    [t]
lfCN Giant    = \ t -> Atom "giant"    [t]
lfCN Wizard   = \ t -> Atom "wizard"   [t]
lfCN Sword    = \ t -> Atom "sword"    [t]
lfCN Dagger   = \ t -> Atom "dagger"   [t]
```

Translating

- Rather than continuing to paste code here, I'll just refer you to the file MCWPL.hs
- Notice that there are two evaluation functions. One, eval, just interprets formulas that involve variables, while the other, evl, evaluates formulas that involve function symbols as well.

Translating sentences

- MCWPL.hs includes three sentences
  - lf1: "Some dwarf defeated some giant"

```
lfSent (Sent (NP1 Some Dwarf)
             (VP1 Defeated (NP1 Some Giant)))
```

  - lf2: "The wizard that Dorothy admired laughed"

```
lfSent (Sent (NP2 The (RCN2 Wizard That Dorothy Admired))
             Laughed)
```

  - lf3: "The princess that helped Alice shuddered

```
lfSent (Sent (NP2 The (RCN1 Princess That (VP1 Helped Alice)))
             Shuddered)
```

Translating sentences

- MCWPL.hs includes sample sentences
- lf1:

"Some dwarf defeated some giant" ⇒

```
Sent (NP1 Some Dwarf) (VP1 Defeated (NP1 Some Giant))

lfSent (...) ⇒
   E x2 conj[dwarf[x2],E x1 conj[giant[x1],defeat[x2,x1]]]
```

i.e.

$$\exists x2\ (dwarf(x2) \wedge \exists x1\ (giant(x1) \wedge defeat(x2, x1)))$$

Translating sentences

- lf2:

  "The wizard that Dorothy admired laughed" ⇒

  Sent (NP2 The (RCN2 Wizard That Dorothy Admired)) Laughed

  lfSent (...) ⇒

  E x1 conj[A x2 (conj[wizard[x2], admire[Dorothy,x2]]<=>x1==x2), laugh[x1]]

  i.e.

  $\exists x1\ (\forall x2\ ((wizard(x2) \wedge admire\ (Dorothy\ (x2))) \leftrightarrow x1 = x2) \wedge laugh(x1))$

Translating sentences

- lf3:

  "The princess that helped Alice shuddered" ⇒

  Sent (NP2 The (RCN1 Princess That (VP1 Helped Alice))

  lfSent (...) ⇒

  E x1 conj[A x2 (conj[princess[x2],help[x2,Alice]]<=>x1==x2), shudder[x1]]

  i.e.

  $\exists x1\ ((\forall x2\ ((princess(x2) \wedge help(x2,Alice)) \leftrightarrow x1 = x2) \wedge shudder(x1))$

Semantics of Predicate Logic

- Now ready to show interpretations in a model.
- See file Model.hs (and Model2.hs) for examples of models of language in FSynF.hs
  - D = {A,B,C,...,Z,Unspec}
    - Because declared as Bounded, can refer to as [minBound..maxBound]
  - Associate constants with elements of D (= Entity)

Model Encoding

- Includes functions to convert from lists to one-place characteristic functions (i.e., for unary relations)
  - Characteristic functions for binary and ternary relations are Curried (e.g., Entity -> Entity -> Bool)
  - *Ignore passivize and self for now.*