The Meaning of Meaning


Philosophical Position:

- Meaning is a mapping from language to the world as experienced by human beings.
- Requires a situated or grounded semantics.

Conventional Approach for NLP:

- Meaning is a mapping from sentences to a meaning representation language.
Meaning Representations

Should be:

- Verifiable.
  - Does Maharani serve vegetarian food?

- Unambiguous.

- Canonical ( = have a Canonical Form).
  - Is vegetarian food served at Maharani?

- Capable of Inference.
  - Can vegetarians eat at Maharani?

- Expressive.

First-Order Logic

```
Formula → AtomicFormula
  | Formula Connective Formula
  | Quantifier Variable, ... Formula
  | ~Formula
  | (Formula)

AtomicFormula → Predicate(Term, ...)

Term → Function(Term, ...)
  | Constant
  | Variable

Connective → ∧ | ∨ | ⊃

Quantifier → ∀ | ∃

Constant → A | VegetarianFood | Maharani...

Variable → x | y | ...

Predicate → Serves | Near | ...

Function → LocationOf | CuisineOf | ...
```
Model-Theoretic Semantics

<table>
<thead>
<tr>
<th>Domain</th>
<th>( \mathcal{D} = { a, b, c, d, e, f, g, h, i, j } )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matthew, Franco, Katie and Caroline</td>
<td>( a, b, c, d )</td>
</tr>
<tr>
<td>Frasca, Med, Rio</td>
<td>( e, f, g )</td>
</tr>
<tr>
<td>Italian, Mexican, Eclectic</td>
<td>( h, i, j )</td>
</tr>
</tbody>
</table>

| Properties | Noisy = \{ e, f, g \} |
| Noisy | Frasca, Med, and Rio are noisy |

| Relations | \( \text{Likes} = \{ (a, f), (c, f), (c, g), (b, e), (d, f), (d, g) \} \) |
| Likes | Matthew likes the Med |
| | Katie likes the Med and Rio |
| | Franco likes Frasca |
| | Caroline likes the Med and Rio |
| Serves | \( \text{Serves} = \{ (e, f), (f, i), (c, h) \} \) |
| Serves | Med serves eclectic |
| | Rio serves Mexican |
| | Frasca serves Italian |

Variables and Quantifiers

Existential Quantifier:

\[ \exists x \left[ \text{Restaurant}(x) \land \text{Serves}(x, \text{MexicanFood}) \land \text{Near} \left( \text{LocationOf}(x), \text{LocationOf}(\text{ICSI}) \right) \right] \]

Universal Quantifier:

\[ \forall x \left[ \text{VegetarianRestaurant}(x) \Rightarrow \text{Serves}(x, \text{VegetarianFood}) \right] \]
Lambda Notation

lambda expression:
\( \lambda x. P(x) \)

lambda reduction:
\( \lambda x. P(x) (A) \rightarrow P(A) \)

lambda expression:
\( \lambda x. \lambda y. \text{Near}(x,y) \)

lambda reduction:
\( \lambda x. \lambda y. \text{Near}(x,y) (\text{Bacaro}) \rightarrow \lambda y. \text{Near}(\text{Bacaro},y) \)
\( \lambda y. \text{Near}(\text{Bacaro},y) (\text{Centro}) \rightarrow \text{Near}(\text{Bacaro},\text{Centro}) \)

States and Events

Consider the following sentences:

- I ate.
- I ate a turkey sandwich.
- I ate a turkey sandwich at my desk.
- I ate at my desk.
- I ate lunch.
- I ate a turkey sandwich for lunch.
- I ate a turkey sandwich for lunch at my desk.

How many arguments should we include in our representation for the predicate “eat”? 
States and Events

One approach (Davidson, 1967):

- Create a variable, \( e \), for the event:
  \[ \exists e\ Eat(e) \]

- Add arguments (perhaps) corresponding to the verb subcategorization frames:
  \[ \exists e\ Eat(e,\ Speaker), \exists e\ Eat(e,\ Speaker,\ TurkeySandwich) \]

- Add separate assertions for everything else:
  \[ \exists e\ Eat(e,\ Speaker) \land Meal(e,\ Lunch) \land Location(e,\ Desk) \land Time(e,\ Tuesday) \]

This is called a Davidsonian representation.

A radical Davidsonian approach (Parsons, 1990) would write everything as a separate assertion:

\[ \exists e\ Eat(e) \land Eater(e,\ Speaker) \land Eaten(e,\ TurkeySandwich) \land Meal(e,\ Lunch) \land Location(e,\ Desk) \land Time(e,\ Tuesday) \]

Is this a good idea?
Consider an “arrive” event:

\[ \exists e \ (\text{Arrive}(e) \land \text{Arriver}(e, \text{Speaker}) \land \text{Destination}(e, \text{NewYork})) \]

To represent distinct tenses:

- I arrived in New York.
  ... \land \text{Precedes}(e, \text{Now})
- I am arriving in New York.
  ... \land \exists i \ (\text{IntervalOf}(e, i) \land \text{MemberOf}(i, \text{Now}))
- I will arrive in New York.
  ... \land \text{Precedes}(\text{Now}, e)

Tenses in English also make use of a reference point (Reichenbach, 1947):

- Past Perfect: I had eaten
  - E \rightarrow R \rightarrow U
- Simple Past: I ate
  - R,E \rightarrow U
- Present Perfect: I have eaten
  - E \rightarrow R,U
- Present: I eat
  - U,R,E
- Simple Future: I will eat
  - U,R,E
- Future Perfect: I will have eaten
  - U \rightarrow E \rightarrow R
Aspect

• Stative expressions:
  - I need the cheapest fare.
  - I want to go first class.
  - * I am needing the cheapest fare on this day.
  - * I carefully like Flight 840 arriving at 10:06.

• Activity expressions:
  - She drove a Mazda.
  - I live in Brooklyn.
  - * She drove a Mazda in an hour.
  - * I live in Brooklyn in a month.

Aspect

• Accomplishment expressions:
  - United flew me to New York.
  - He booked me a reservation.
  - She stopped booking my flight.
  - She booked a flight in a minute.

• Achievement expressions:
  - She found her gate.
  - I reached New York.
  - * I reached New York for a few minutes.
Semantic Analysis

Let's take a simple pipeline approach:

1. Inputs → Syntactic Analysis → Syntactic Structures → Semantic Analysis → Meaning Representations

Semantic Analysis

Let's apply the principle of compositionality:

How can we get this to work?
What is the “meaning” of each component?
Semantic Attachment

Sentence: Maharani closed.

In English, determiners have narrow scope:

\[ [S \ [NP [Det Every] [Nom [N restaurant]]] [VP [V closed]]] \]

In First-Order Logic, quantifiers have wide scope:

\[ \forall x [ Restaurant(x) \Rightarrow \exists e (Closed(e) \land ClosedThing(e, x))] \]

How can we translate from one to the other?
Semantic Attachment

Sentence: Every restaurant ...

\[
NP \rightarrow \text{Det Nominal} \\
\{ \text{Det.sem(Nominal.sem)} \} \\
\lambda Q. \forall x [\lambda x. \text{Restaurant}(x) \Rightarrow Q(x)] \\
\lambda Q. \forall x [\text{Restaurant}(x) \Rightarrow Q(x)]
\]

\[
\text{Det} \rightarrow \text{Every} \\
\{ \lambda P. \lambda Q. \forall x [P(x) \Rightarrow Q(x)] \}
\]

\[
\text{Nominal} \rightarrow \text{Noun} \\
\{ \text{Noun.sem} \}
\]

\[
\text{Noun} \rightarrow \text{restaurant} \\
\{ \lambda x. \text{Restaurant}(x) \}
\]

Semantic Attachment

Sentence: Every restaurant closed.

\[
S \rightarrow NP \ VP \\
\{ NP.sem(VP.sem) \} \\
\forall x [\text{Restaurant}(x) \Rightarrow \lambda y. \exists e [\text{Closed}(e) \land \text{ClosedThing}(e, y)](x)] \\
\forall x [\text{Restaurant}(x) \Rightarrow \exists e [\text{Closed}(e) \land \text{ClosedThing}(e, x)]]
\]

\[
NP \rightarrow \text{Det Nominal} \\
\{ \text{Det.sem(Nominal.sem)} \} \\
\lambda Q. \forall x [\text{Restaurant}(x) \Rightarrow Q(x)]
\]

\[
\text{VP} \rightarrow \text{Verb} \\
\{ \text{Verb.sem} \}
\]

\[
\text{Verb} \rightarrow \text{closed} \\
\{ \lambda y. \exists e [\text{Closed}(e) \land \text{ClosedThing}(e, y)] \}
\]
Semantic Attachment

Sentence: *Every restaurant closed.*

Let's check again: *Maharani closed.*
And let's try: *Matthew opened a restaurant.*
Semantic Attachment

Sentence: Matthew opened a restaurant.

The semantic attachment for existential quantifiers should be:

\[
\text{Det} \rightarrow \text{a} \\
\{\lambda P. \lambda Q. \exists x [P(x) \land Q(x)]\}
\]

If \( VP \rightarrow \text{Verb NP} \) is interpreted as \( \{\text{Verb.sem}(\text{NP.sem})\} \), then for the transitive verb opened we could write:

\[
\text{Verb} \rightarrow \text{opened} \\
\{\lambda W. \lambda z. W(\lambda y. \exists e[\text{Opened}(e) \land \text{Opener}(e, z) \land \text{OpenedThing}(e, y)]\)}
\]

This works! But is this the best way to do things?

<table>
<thead>
<tr>
<th>Grammar Rule</th>
<th>Semantic Attachment</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S \rightarrow \text{NP VP} )</td>
<td>( {\text{NP.sem}(\text{VP.sem})} )</td>
</tr>
<tr>
<td>( \text{NP} \rightarrow \text{Det Nominal} )</td>
<td>( {\text{Det.sem}(\text{Nominal}.sem)} )</td>
</tr>
<tr>
<td>( \text{NP} \rightarrow \text{ProperNoun} )</td>
<td>( {\text{ProperNoun}.sem} )</td>
</tr>
<tr>
<td>( \text{Nominal} \rightarrow \text{Noun} )</td>
<td>( {\text{Noun}.sem} )</td>
</tr>
<tr>
<td>( \text{VP} \rightarrow \text{Verb} )</td>
<td>( {\text{Verb}.sem} )</td>
</tr>
<tr>
<td>( \text{VP} \rightarrow \text{Verb NP} )</td>
<td>( {\text{Verb}.sem(\text{NP}.sem)} )</td>
</tr>
<tr>
<td>( \text{Det} \rightarrow \text{every} )</td>
<td>( {\lambda P. \lambda Q. \exists x [P(x) \Rightarrow Q(x)]} )</td>
</tr>
<tr>
<td>( \text{Det} \rightarrow \text{a} )</td>
<td>( {\lambda P. \lambda Q. \exists x [P(x) \land Q(x)]} )</td>
</tr>
<tr>
<td>( \text{Noun} \rightarrow \text{restaurant} )</td>
<td>( {\lambda r. \text{Restaurant}(r)} )</td>
</tr>
<tr>
<td>( \text{ProperNoun} \rightarrow \text{Matthew} )</td>
<td>( {\lambda m. \text{m(Matthew)}} )</td>
</tr>
<tr>
<td>( \text{ProperNoun} \rightarrow \text{Franco} )</td>
<td>( {\lambda f. \text{f(Franco)}} )</td>
</tr>
<tr>
<td>( \text{ProperNoun} \rightarrow \text{Francia} )</td>
<td>( {\lambda f. \text{f(Francia)}} )</td>
</tr>
<tr>
<td>( \text{Verb} \rightarrow \text{closed} )</td>
<td>( {\lambda x. \exists e \text{Closed}(e) \land \text{ClosedThing}(e, x)} )</td>
</tr>
<tr>
<td>( \text{Verb} \rightarrow \text{opened} )</td>
<td>( {\lambda w. \lambda z. w(\lambda x. \exists e \text{Opened}(e) \land \text{Opener}(e, z) \land \text{Opened}(e, x))} )</td>
</tr>
</tbody>
</table>
Semantics in PROLOG

wwdcg0.pl:

sentence(S) -->
    noun_phrase(Person, Number, VP^S),
    verb_phrase(Person, Number, VP).

verb_phrase(Person, Number, X^S) -->
    verb(Person, Number, [pps], X^Y),
    prepositional_phrases(Y^S).

noun_phrase(3, singular, NP) --> proper_noun(NP).
noun_phrase(3, Number, NP) -->
    determiner(Number, CNP^NP),
    common_noun_phrase(Number, CNP).

determiner(Number, LF) --> [Word], {det(Word, Number, LF)}.

det(every, singular, (X^S1)^S2^ forall(X,S1=>S2)).
det(a    , singular, (X^S1)^S2^ exists(X,S1&S2)).
det(some , singular, (X^S1)^S2^ exists(X,S1&S2)).
det(the  , singular, (X^S1)^S2^ unique(X,S1&&S2)).
det(all  , plural,   (X^S1)^S2^ forall(X,S1=>S2)).
det(the  , plural,   (X^S1)^S2^ forall(X,S1=>S2)).
det(some , plural,   (X^S1)^S2^ exists(X,S1&S2)).
Semantics in PROLOG

Simple semantics, without quantifiers:

Let's encode this using PROLOG unification.

Notation:
To encode $\lambda x.\lambda y.\text{wrote}(y,x)$ use: $X^Y^\text{wrote}(Y,X)$.

Logical Form:
$s(S) \rightarrow \text{np}(\text{NP}), \text{vp}(\text{VP})$.

Lambda reduction:
Define:
reduce(Arg^Expr, Arg, Expr).

Then to reduce $\lambda x.\text{halts}(x)(\text{shrdlu})$ call:
| ?- reduce(X^\text{halts}(X), \text{shrdlu}, LF).
| $X$ = \text{shrdlu},
| $LF$ = \text{halts}(\text{shrdlu}) ?
Semantics in PROLOG

Grammar with logical forms and lambda reduction:

\[
\begin{align*}
 s(S) & \rightarrow \text{np}(\text{NP}), \text{vp}(\text{VP}), \text{reduce}(\text{VP}, \text{NP}, S).
\text{vp}(\text{VP}) & \rightarrow \text{tv}(\text{TV}), \text{np}(\text{NP}), \text{reduce}(\text{TV}, \text{NP}, \text{VP}).
\text{vp}(\text{VP}) & \rightarrow \text{iv}().
\text{tv}(X^Y^wrote(Y,X)) & \rightarrow \text{[wrote]}. \\
\text{iv}(X^halts(X)) & \rightarrow \text{[halts]}. \\
\text{np}(shrdlu) & \rightarrow \text{[shrdlu]}.
\text{np}(terry) & \rightarrow \text{[terry]}. \\
\end{align*}
\]

Grammar with partial execution of reduce:

\[
\begin{align*}
 s(S) & \rightarrow \text{np}(\text{NP}), \text{vp}(\text{NP}^\text{S}).
\text{vp}(\text{VP}) & \rightarrow \text{tv}(\text{NP}^\text{VP}), \text{np}(\text{NP}).
\end{align*}
\]

Semantics in PROLOG

Parsing with semantic interpretation:

\[
\begin{align*}
| ?- s(LF, [shrdlu, halts], []). & \\
LF = \text{halts}(\text{shrdlu}) ? & \text{yes} \\
| ?- s(LF, [terry, wrote, shrdlu], []). & \\
LF = \text{wrote}(\text{terry}, \text{shrdlu}) ? & \text{yes}
\end{align*}
\]

Now let's modify the grammar to include quantified noun phrases.
Semantics in PROLOG

Universal quantifiers:
\[ \lambda P . \lambda Q . \forall x [P(x) \Rightarrow Q(x)] \]
\[ (X^P)^{(X^Q)} \forall (X, (P \Rightarrow Q)) \]

Existential quantifiers:
\[ \lambda P . \lambda Q . \exists x [P(x) \land Q(x)] \]
\[ (X^P)^{(X^Q)} \exists (X, (P \land Q)) \]

Lexical entries:
\[
\text{det} \{- \} \rightarrow [D], \{\text{det}(D, LF)\}.
\text{det}(\text{every}, (X^P)^{(X^Q)} \forall (X, (P \Rightarrow Q))).
\text{det}(a, (X^P)^{(X^Q)} \exists (X, (P \land Q))).
\]

Semantics in PROLOG

For noun phrases, apply Det.sem to Noun.sem:
\[ \text{np}(NP) \rightarrow \text{det}(N^NP), n(N). \]

For sentences, apply NP.sem to VP.sem:
\[ s(S) \rightarrow \text{np}(VP^S), \text{vp}(VP). \]

If nouns and intransitive verbs have semantics:
\[ n( X^\text{program}(X) ) \rightarrow [\text{program}]. \]
\[ iv( X^\text{halts}(X) ) \rightarrow [\text{halts}]. \]

then:
\[ ?- s(LF, [\text{every, program, halts}], []). \]
\[ LF = \forall _A \text{program}(_A) \Rightarrow \text{halts}(_A) ? \]
\[ yes \]
Semantics in PROLOG

For transitive verbs, we want the phrase “wrote a program” to have the same semantic form as the phrase “halts”:

\[ \lambda z. \exists p[p \land wrote(z, p)] \]

Let’s do this by applying NP.sem to Verb.sem:

\[
vp(Z^S) \rightarrow tv(TV), np(NP), \{reduce(TV, Z, IV), reduce(NP, IV, S)\}.
\]

With partial execution, this becomes:

\[
vp(Z^S) \rightarrow tv(Z^IV), np(IV^S).
\]

Adding another noun and a transitive verb:

\[
n( X^\text{student}(X) ) \rightarrow [\text{student}].
\]

\[
tv( X^Y^\text{wrote}(Y, X) ) \rightarrow [\text{wrote}].
\]

we can parse the following sentence:

```
| ?- s(LF, [every, student, wrote, a, program], []).
LF = forall(_A,student(_A)
   => exists(_B,program(_B)
       &
       wrote(_A, _B))) ?
yes
```
Problems with Logical Forms

The quantifier mapping is very complex.

- Narrow scope in English, wide scope in logic.

The quantifier scope is fixed.

\[ \exists A \text{ wumpus is in every pit.} \]
\[ \exists A \text{ wumpus}(A) \]
\[ \land \forall B \text{ pit}(B) \]
\[ \Rightarrow \text{ in}(A, B) \}

But this sentence can be interpreted in two ways:

\[ \exists A \left[ \text{Wumpus}(A) \land \forall B \left[ \text{Pit}(B) \Rightarrow \text{In}(A, B) \right] \right] \]
\[ \forall B \left[ \text{Pit}(B) \Rightarrow \exists A \left[ \text{Wumpus}(A) \land \text{In}(A, B) \right] \right] \]

Quasi-Logical Forms

\[ S(\exists e \in \text{Perceive}(\forall \text{Agent}(a)), \{\exists w \text{ Wumpus}(w)\}, \text{Nose}) \]
\[ \land \text{During}(\text{Now}, e) \]

\[ \forall \text{Agent}(a) \]
\[ \text{NP}(\{\exists w \text{ Wumpus}(w)\}) \]

\[ \text{NP}(\forall \text{Agent}(a)) \]
\[ \text{NP}(\forall \text{Agent}(a)) \]

\[ \text{Article}(\forall) \text{ Noun}(\text{Agent}) \]
\[ \text{Article}(\forall) \text{ Noun}(\text{Agent}) \]

\[ \text{Every agent smells a wumpus} \]
Scaling Up to Real Sentences


<table>
<thead>
<tr>
<th></th>
<th>cases</th>
<th>words</th>
<th>sentences</th>
<th>parse trees</th>
<th>null parses</th>
</tr>
</thead>
<tbody>
<tr>
<td>SupCt May 1999</td>
<td>9</td>
<td>35,796</td>
<td>1,213</td>
<td>1,141</td>
<td>5.94%</td>
</tr>
<tr>
<td>2dCirc May 1999</td>
<td>31</td>
<td>66,383</td>
<td>2,941</td>
<td>2,710</td>
<td>7.50%</td>
</tr>
<tr>
<td>June 1999</td>
<td>26</td>
<td>108,778</td>
<td>3,683</td>
<td>3,600</td>
<td>4.56%</td>
</tr>
<tr>
<td>3dCirc May 1999</td>
<td>15</td>
<td>71,088</td>
<td>2,443</td>
<td>2,286</td>
<td>4.68%</td>
</tr>
<tr>
<td>June 1999</td>
<td>30</td>
<td>148,820</td>
<td>5,382</td>
<td>5,132</td>
<td>4.66%</td>
</tr>
<tr>
<td>Total</td>
<td>111</td>
<td>430,265</td>
<td>15,962</td>
<td>14,529</td>
<td>5.42%</td>
</tr>
</tbody>
</table>

She has also brought this ADA suit in which she claims that her former employer, Policy Management Systems Corporation, discriminated against her on account of her disability: 526 U.S. 795 (1999)
She has also brought this ADA suit in which she claims that her former employer, Policy Management Systems Corporation, discriminated against her on account of her disability.

526 U.S. 795 (1999)

**Ontological Promiscuity**

Every relationship is an object:

\( \text{Own} \ O \ (\text{Actor} \ A) \ (\text{Property} \ P) \)

"O represents an instance of ownership"

Every object is a relationship:

\( \text{Arch} \ A \ (\text{Lintel} \ T) \ (\text{Column} \ L) \ (\text{Column} \ R) \)

"T, L, R are in the arch relation to one another"
"She has also brought this ADA suit ..."
Terms:
\( \text{nterm}(\text{lex}, \text{var}, \text{list}) \)
\( \text{nterm}(\text{suit}, D, []) \)
\( \text{nterm}(\text{she}, G, []) \)
\( \text{nterm}(\text{employer}, J, []) \)
\( \text{nterm}(\text{Corporation}, N, []) \)
\( \text{nterm}(\text{her}, P, []) \)
\( \text{nterm}(\text{account}, R, []) \)
\( \text{nterm}(\text{disability}, T, []) \)

Terms:
\( \text{aterm}(\text{lex}, \text{var}, \text{list}) \)
\( \text{aterm}(\text{former}, I, [J]) \) & \( \text{nterm}(\text{employer}, J, []) \)
\( \text{sterm}(\text{brought}, A, []) \)
& \( \text{aterm}(\text{also}, U, [A]) \)
Terms:
- `pterm(lex, var, list)`
- `pterm(in, F, [E, _])`
- `pterm(against, O, [H, _])`
- `pterm(on, Q, [H, _])`
- `pterm(of, S, [R, _])`

Two Operations:
1. An expression can be embedded in the list of constituents of a term.
2. An expression can be adjoined either before or after a term.
Determiners:

- `nterm(suit,D,[])` /det(this,nn)
- `nterm(she,G,[])` /det(null,prp)
- `nterm(employer,J,[])` /det(her,nn)
- `nterm(Corporation,N,[])` /det(null,nnp)
- `nterm(disability,T,[])` /det(her,prp)

Tense and Aspect:

- `sterm(brought,A,[_,_])` /present,perfect
- `sterm(claims,E,[_,_])` /present
- `sterm(discriminated,H,[_])` /
Variables:

\[
\text{sterm}(\text{discriminated, } H, [\_]) \land \text{opterm}(\text{against, } O, [H, \_]) \land \text{qterm}(\text{on, } Q, [H, \_])
\]

Variables:

\[
\text{nterm}(\text{suit, } D, []) \land \text{Fterm}(\text{in, } F, \\
\text{Eterm}(\text{which, } D, [])) \land \text{sterm}(\text{claims, } E, [\_])
\]
The petitioner contends that the regulatory takings claim should not have been decided by the jury and that the Court of Appeals adopted an erroneous standard for regulatory takings liability.

526 U.S. 687 (1999)

The court ruled that sufficient evidence had been presented to the jury from which it reasonably could have decided each of these questions in Del Monte Dunes' favor.

526 U.S. 687 (1999)
Definite Clause Grammars

- Constructed by hand, with automated tools:
  - Tally all the grammar rules that appear in the parsed legal corpus, in three variants:
    - Main grammar rules, with distinct heads and complements.
    - Modifier rules, with a base category and a nonterminal premodifier.
    - Modifier rules, with a base category and a nonterminal postmodifier.
  - Retrieve any fragment of any parse tree, anywhere in the corpus.
  - Write more expressive grammars for particular constructs (e.g., for English auxiliary verb system).
- Approximately 700 rules.

Semantics of noun phrase: \( X^\text{nterm}(\text{she}, X, []) \)
Semantics of verb phrase: \( \text{Subj}^E^\text{sterm}(\text{brought}, E, [\text{Subj}, _]) \)

Unify:
- \( \text{NP} = \text{nterm}(\text{she}, X, []) \)
- \( \text{Det} = \text{det}(\text{null}, \text{prp}) \)
- \( \text{Subj} = \text{NP}/\text{Det} \)
- \( \text{Term} = \text{sterm}(\text{brought}, E, [\text{Subj}, _]) \)

Result:
- \( E^*\text{Term} = \text{sterm}(\text{brought}, E, [\text{nterm}(\text{she}, X, [])/\text{det}(\text{null}, \text{prp}), _]) \)
Semantics of 'WDT' and 'WHNP': $W^n\text{term}(\text{which}, W, [])$

Semantics of 'IN': Obj $^*$ Subj $^*$ P $^*$ term(in, P, [Subj, Obj])

Unify: Obj $\rightarrow$ nterm(\text{which}, W, [])

Term $\rightarrow$ pterm(in, P, [Subj, Obj])

Semantics of 'WHPP':

$W^n\text{Subj}\text{--P}\text{term}(in, P, [\text{Subj}, \text{nterm}(\text{which}, W, [])])$

Semantics of 'S': $E^*\text{term}(\text{claims}, E, [\_, \_])$

Unify: Term $\rightarrow$ pterm(in, P, [E, nterm(\text{which}, W, [])])

Tense $\rightarrow$ [present]

Semantics of 'SBAR':

$W^n(E^*\text{term}(in, P, [E, nterm(\text{which}, W, [])]) \&
\text{term}(\text{claims}, E, [\_, \_])/[\text{present}])$
Carter & Carter v. Exxon USA, Ct. App. 3d Circ. (May 24, 1999):

Richard Carter and his wife, Carol, appeal and argue that the district court erred in granting summary judgment in favor of Exxon Company USA on their Petroleum Marketing Practices Act claim and on Exxon’s state law counterclaim.

Here is the semantic interpretation (qlf) of this sentence.