6.863J Natural Language Processing
Lecture 19: the meaning of it all, #5

Instructor: Robert C. Berwick
berwick@ai.mit.edu

The Menu Bar

- Administrivia:
  - Lab 4(a&b) out April 16—last lab before final project

Agenda:
  - Being curteous: from meaning to discourse
  - How to use language
The story so far

- We can map (english) language to lambda formulas
- We can use FOL to check them
- We can use model theory to see if they can be satisfied

- How does this fit in..?

The Language use domain

- As inference tasks: (cf press conference)
  - Querying
  - Consistency checking
  - Informativity checking (why?)
Querying

- Given a model $M$ and a formula $\phi$, is $\phi$ true in model $M$ or not?
- $M$ is a little picture of the world (e.g., inside Bush’s brain...)
- Querying $\phi$ is asking whether or not the info is true in this little piece
- We need a model checker for this
- For finite models – easy to do, and needed for question answering

Consistency checking

- A formula is consistent if it is satisfiable in at least one model - such formulas describe ‘conceivable’ or ‘possible’ states of affairs. E.g., silly(bob) is consistent
- A formula that is not consistent is inconsistent eg, silly(bob) & not silly(bob)
- A finite set of formulas is consistent if its conjunction is consistent, otherwise, inconsistent
Consistency checking

- We would like to do this – why?
- If inconsistent information, something might be going wrong with communication in discourse
- But this is much harder to check...
- It is undecidable!
- We have to use model builder and thm prover to at least help

Informativity

- A valid sentence is a sentence that is true in all models (e.g., \( \text{silly(bob)} \lor \neg \text{silly(bob)} \)). A sentence that is not valid is invalid.
- Formula set \( \Phi \), and new formula \( \varphi \)
- Valid argument: formula set \( \Phi \) implies \( \varphi \) (in all models)
- Invalid argument: otherwise
Informativity

- Valid sentence is **uninformative** Why?
- Doesn’t give us any specific information (true in all possible models)
- A sentence that is not valid is **informative**
- Otherwise, **uninformative**
- (wrt to some collection of formulas...)

Informativity

- Also harder than querying
- Undecidable for FOL
Informativity and consistency

- If \( \phi \) informative = not valid = iff not \( \phi \) is valid, so the opposite of \( \phi \) really was an option
- Contrariwise, if \( \phi \) uninformative then not \( \phi \) is invalid, so the opposite of \( \phi \) is not an option
- So, we can use a theorem prover to kill two birds with one stone (is that an idiom?)

Theorem prover

- Used to tell us whether a formula is valid or not
- Proof theory: purely *syntactic* way to figure out whether a formula is valid or not
- Methods (see AI) – tableaux and resolution theorem proving
- Try to prove the negation of the formula – if you can’t, then the formula is valid
- If we have premises true and a result false, then informative (negation of (\( \phi \) implies \( \varphi \)))
What happens if theorem prover doesn’t get answer?

- FOL undecideable
- So, if no answer, don’t know if the formula is not a theorem... (is not valid)
- If there is an answer, pretty sure the formula is a theorem (is valid)

Model building

- Theorem provers check whether a formula or set of formulas is valid (true in all possible models)
- Model builders attempt to construct a formula (or set of formulas) and so show that this formula is satisfiable (true in at least one possible model)
- So – must limit model builders to domain size...
- Uncertainty: if you don’t find model, you don’t know... but
- If you do, pretty sure the formula is satisfiable
- Restricted to finite models (Everybody has a mother, even George Bush)
Theorem proving and model building

Consistency
To check whether $\phi$ is consistent...
- Give $\neg\phi$ to a theorem prover; if it finds a proof, $\phi$ is not consistent
- Give $\neg\phi$ to a model builder; if it finds a model, then $\phi$ is consistent

Informativity
To check whether $\phi$ is informative wrt $\psi$:
- Give $\psi \rightarrow \phi$ to a theorem prover; if it finds a proof, $\phi$ is not informative wrt $\psi$
- Give $\psi \land \phi$ and $\psi \land \neg\phi$ to a model builder; if it finds a model in both cases, then $\phi$ is informative wrt $\psi$
The Bob hierarchy

- Dumb bob - just parse and quantifier assignment, no inferences
- Clever bob – only consistent inferences (logical syntax only...)

The bob hierarchy

- Mia smokes and does not smoke
- Bob: OK
- Vincent likes every woman
- Bob: OK
- Mia is a woman; Vincent does not like Mia
- Bob: OK
Clever Bob

- Use model builder mace to check consistency, and a theorem prover otter to check inconsistency
- Use this to reject inconsistent sentences

Representing Discourse

- Discourse so far: a collection of the previous sentences = D
- Add single new sentence, φ.
- Does D imply ¬φ (in all models)?
- If so, then φ is inconsistent
Actual program: add “consistency”

```prolog
curtUpdate(Input,Moves,run):-
kellerStorage(Input,Readings), !,
updateHistory(Input),
readings(OldReadings),
combine(Readings,OldReadings,Combined),
consistentReadings(Combined-NewReadings,Moves),
updateReadings(NewReadings),
updateModels(Models).
```

Rugrat bob

- Mia is a woman
- Vincent likes every woman
- Vincent does not like Mia
- Must be able to do equality reasoning:
  \[
  \text{woman}(A) \land \text{mia} = A
  \]

- Need to do general theorem proving...but this can be hard...
- Solution:
Clever Bob

- Run model builder and theorem prover in parallel
- Why?
  - If a discourse is inconsistent, then a theorem prover will never be able to detect an inconsistency - just runs until clock's up (negative test for consistency – are no WMD in Iraq)
  - Model builder is a **positive** check for consistency

---

Clever bob

> Mia dances.

Message (consistency checking): mace found a result. Curt: OK.

> models

```
1 model([d1],[f(1,dance,d1),f(0,mia,d1)])
```
Models not only what you might expect... Why?

> Jody dances

Message (consistency checking): mace found a result.
Curt: OK.

> models

1 model([d1],[f(1,dance,[d1]),f(0,jody,d1),f(0,mia,d1)]

It doesn’t know otherwise...

> Mia is not Jody.

Message (consistency checking): mace found a result.
Curt: OK.

> models

1 model([d1,d2],[f(0,mia,d1),f(0,jody,d2),f(1,dance,[d1,d2])]
Both thm prover & model builder

- Vincent is a man
  - Consistency – mace finds result
- Mia likes every man
  - Consistency – mace
- Mia does not like Vincent
  - Doesn’t believe it – uses thm prover

Informativeness

- Theorem prover gives negative check for informativeness – if Discourse-so-far implies the new sentence $\phi$ (as a theorem) the new sentence $\phi$ is uninformative
- Model builder gives positive check for informativeness – if model builder can show that Discourse-so-far $\cup \{ \neg \phi \}$ has a model, then latest sentence is informative
Example

- Vincent knows every boxer
- Butch is a boxer
- (therefore) Vincent knows Butch – valid
vs...
- If Vincent snorts then Jody smokes
- Jody smokes
- Vincent snorts – what will it say? What about Vincent does not snort

Can we use consistency check for informativeness?

- Consistency done first – so $\phi$ known to be consistent with previous discourse
- Suppose M is the model made so far
- Suppose new sentence $\phi$ is false in this model M
- What does this tell us? Is $\phi$ informative?
Eliminating logical duplicates

- A boxer loves a woman
- Has two readings from quantifiers, and two model results:

```
> readings
1 exists A (boxer(A) & exists B (woman(B) & love(A,B)))
2 exists A (woman(A) & exists B (boxer(B) & love(B,A)))
```

What about this one?

- Every boxer loves a woman
  - System as it stands says two readings “probably now equivalent” (theorem prover)
  - Why? Can’t we do better?
  - What about having the strongest reading only? What else to cut down on thm proving burden?
If ignorance is bliss

- Knowledgeable Curt
- Use background knowledge as additional premises
- Add lexical knowledge and world knowledge

Consistency & Informativeness

- Consistency now:
  - [negative test] Lexical knowledge $\cup$ World knowledge $\cup$ Discourse-so-far $\not\models \phi$
  - [positive test] Lexical knowledge $\cup$ World knowledge $\cup$ Discourse-so-far $\cup\{\phi\}$ has a model
Informativeness

- [Negative test] Lexical knowledge $\cup$ World knowledge $\cup$ Discourse-so-far $\Rightarrow \phi$

- [positive test] Lexical knowledge $\cup$ World knowledge $\cup$ Discourse-so-far $\cup \{\neg \phi\}$ has a model

So let’s see what this does

- Mia smokes gives us: smoke(mia)
- What does this take?
(forall A (concrete(A) > ~ abstract(A)) &
(forall B (entity(B) > concrete(B)) &
(forall C (entity(C) > thing(C)) &
(forall D (living(D) > ~ nonliving(D)) &
(forall E (male(E) > ~ female(E)) &
(forall F (organism(F) > living(F)) &
(forall G (organism(G) > entity(G)) &
(forall H (animate(H) > ~ inanimate(H)) &
(forall I (human(I) > ~ nonhuman(I)) &
(forall J (person(J) > human(J)) &
(forall K (person(K) > animate(K)) &
(forall L (person(L) > organism(L)) &
(forall M (female(M) > ~ male(M)) &
(female(mia) & (person(mia) &
(female(mia) & person(mia))))))))))))))))

Hypernym ('above')

Hyponym ('below')

Not transitive!
Hypernym: All X, car x implies vehicle x
All x, concrete x implies not abstract x
World knowledge

- Only persons can dance
- For all x, Dance(x) implies person(x)

- drink: For all x, for all y, drink(x,y) implies person(x) & beverage(y)

- Plays into consistency and in rejecting scope readings: 'Every car has a radio'

World knowledge helps...

> ?- readings

1 forall A (car(A) > exists B (radio(B) & have(A, B)))
forall T forall U (exists V
  (object(T) & (object(U) & (object(V) &
   (have(T, V) & have(U, V))))) > T = U))

> 1 car... (compare: every boxer has a broken nose)
Helpful bob

- Vincent likes Mia
- Who likes a plant?
- Ans: “I have no idea”
- Answering questions – yes, no, or no answer...
- Query model builder with free variable for x, corresponding to ‘who’

How it’s done

Models=[Model|_],
satisfy(some(X,and(R,S)),Model,[],Result),
\+ Result=undef,
!,
findall(A,satisfy(and(R,S),Model,[g(X,A)],pos),Answers),
realiseAnswer(Answers,que(X,R,S),Model,String),
Moves=[sensible_question,answer(String)]
;
Moves=[unknown_answer]
).
Is this all for answering a discourse query?

• No!
• Consider: discourse models show a possible picture of the world – the way the agent imagines them to be, not necessarily the way things are
• What can go wrong?
• Example: Mia or Jody dances. Who dances?
• If just say: Mia, or just Jody – this is more restrictive...

What to do?

• Check whether answer is just possible or whether the answer is guaranteed... by using theorem prover on what model builder has selected

• Jody or Mia dances (dance(J) OR dance(M))
• Build model in which “Jody dances” is true
• ‘Who dances’ finds ‘Jody’ as candidate answer – but perhaps this is so because of discourse..does this answer follow logically from discourse so far & background knowledge?
Answer

- Try to prove dance(J) from background & discourse alone
- Won’t work – it’s a disjunction
- So, hedge bet

Generating answers

- Even a bit of discourse/communication here
- Why do we answer ‘Jody’ instead of ‘a person’?

- Generating more specific answers – when? How?
- We need a theory!
Discourse representation theory (DRT)

- Semantic framework w/ a language to describe discourse
- Translate discourse to FO logic
- Compatible with lambda calculus approach

DRT overview

- Uses language based on box-like structures called DRSs (discourse representation structures)
- Intuition: DRSs are pictures
- Another (nonrepresentational) view: DRSs are programs
Discourse

- Mia is a woman. She loves Vincent
- A man snorts. He collapses.
- Problems: complex post-processing & counter-intuitive readings

We will see if we can do this..

- If a criminal eats a big kahuna burger, he enjoys it
- Translation – the correct one – is:
  \[ \forall x \forall y \left[ \text{criminal}(x) \land \text{big}_k\_b(y) \land \text{eat}(x,y) \rightarrow \text{enjoy}(x,y) \right] \]

But our system current gets:

\[ \exists x \left[ \text{criminal}(x) \land \exists y \left[ \text{big}_k\_b(y) \land \text{eat}(x,y) \right] \rightarrow \text{enjoy}(x,y) \right] \]
Context change potential

- When we utter ‘a man snorts’ we don’t simply make a claim about the world, we change the context in which subsequent utterances will be interpreted (hmm, like a frame….)
- Start a new discourse with the empty box

Changing context

- Start a new discourse with the empty box

- Expand this box with info from the entire discourse

man(x)

snorts(x)
Pronouns

A man snorts. He collapses

1. Add new discourse referent, y
2. Add condition ‘collapse(y)’
3. Add equation ‘x=y’

<table>
<thead>
<tr>
<th>x, y</th>
</tr>
</thead>
<tbody>
<tr>
<td>man(x)</td>
</tr>
<tr>
<td>snorts(x)</td>
</tr>
<tr>
<td>collapse(y)</td>
</tr>
<tr>
<td>x=y</td>
</tr>
</tbody>
</table>

The discourse referent introduced must be identified with an accessible discourse referent

Discourse 2

Vincent snorts. He collapses.

<table>
<thead>
<tr>
<th>x, y</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=Vincent</td>
</tr>
<tr>
<td>snorts(x)</td>
</tr>
<tr>
<td>collapse(y)</td>
</tr>
<tr>
<td>y=x</td>
</tr>
</tbody>
</table>

Same as quantified NPs...equational
DRT summary so far

- Pictures of changing context
- By introducing discourse referents and stating constraints
- Proper names and quantified NPs handled the same
- Parallel between anaphoric NPs and proper names

DRS languages

- Handle universal quantification and negation
- DRSs nested, combined with connectives
  DRS languages like FOL
  - Contain connectives ∨, ¬, →, = (but not usually ∧)
  - Symbols x, y, z, ... - these are called discourse referents, not variables
Differences
  - Don’t contain ∀ or ∃ (this is done by boxes for ∀ or implicit, for ∃)
Examples

- We’ve seen indefinite NP, ‘a man snorts’, proper name, eg, Vincent does not snort

\[
\begin{array}{c|c|c}
\hline
x & x = \text{vincent} & \neg \text{snort}(x) \\
\hline
\end{array}
\]

Universal quantifiers

- Every boxer snorts

\[
\begin{array}{c|c|c}
\hline
x & \text{boxer}(x) & \text{snort}(x) \\
\hline
\end{array}
\]
Informal semantics for DRS

- Q: When is a DRS satisfied in a model?
- A: Iff it is an accurate image of the info recorded inside the model

<table>
<thead>
<tr>
<th>x, y</th>
</tr>
</thead>
<tbody>
<tr>
<td>woman(x)</td>
</tr>
<tr>
<td>boxer(x)</td>
</tr>
<tr>
<td>admire(x, y)</td>
</tr>
</tbody>
</table>

Complex conditions

- Negated DRS: satisfied if it is not possible to embed the picture inside the model
- Disjunctive: can embed both parts in model
- Implicational: no matter what entities used to embed antecedent, we can embed consequent
Most important constraint - referents

- **Accessibility**: a geometric concept – the way DRSs are stacked inside one another

- Discourse referents of DRS K1 are accessible from DRS K2 when K1 equals K2 or when K1 subordinates K2
- Intuitively: look up and then look left (with $\rightarrow$)

Calculating accessibility

- Vincent snorts. He collapses.

<table>
<thead>
<tr>
<th>x, y</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=Vincent snorts(x) collapse(y) y=x</td>
</tr>
</tbody>
</table>

- x is accessible to y (they are part of the same DRS)
Calculating accessibility

- Every boxer snorts. He collapses.

<table>
<thead>
<tr>
<th>x</th>
<th>snort(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>boxer(x)</td>
<td></td>
</tr>
<tr>
<td>collapse(y)</td>
<td></td>
</tr>
<tr>
<td>y=?</td>
<td></td>
</tr>
</tbody>
</table>

Back to the Kahuna burger...

- How do we represent this in DRT?
Questions

- Does the DRS representation really capture the meaning?
- Can we build the representations systematically?
- A: Yes, we can translate to FOL and get the right answer...
- A: Yes, you can do it top down or bottom up