Meaning Representation and Semantic Analysis

Ling 571
Deep Processing Techniques for NLP
February 9, 2011

Roadmap

- Meaning representation:
  - Event representations
  - Temporal representation
- Semantic Analysis
  - Compositionality and rule-to-rule
  - Semantic attachments
    - Basic
    - Refinements
  - Quantifier scope
  - Earley Parsing and Semantics

FOL Syntax Summary

$$
\begin{array}{rcl}
\textit{Formula} & \rightarrow & \textit{AtomicFormula} \\
& | & \textit{Formula Connective Formula} \\
& | & \textit{Quantifier Variable}, \ldots \textit{ Formula} \\
& | & \neg \textit{ Formula} \\
& | & (\textit{Formula}) \\
\textit{AtomicFormula} & \rightarrow & \textit{Predicate}(\textit{Term}, \ldots) \\
\textit{Term} & \rightarrow & \textit{Function}(\textit{Term}, \ldots) \\
& | & \textit{Constant} \\
& | & \textit{Variable} \\
\textit{Connective} & \rightarrow & \wedge \mid \vee \mid \Rightarrow \\
\textit{Quantifier} & \rightarrow & \forall \mid \exists \\
\textit{Constant} & \rightarrow & A \mid \textit{VegetarianFood} \mid \textit{Maharani} \cdots \\
\textit{Variable} & \rightarrow & x \mid y \mid \cdots \\
\textit{Predicate} & \rightarrow & \textit{Serves} \mid \textit{Near} \mid \cdots \\
\textit{Function} & \rightarrow & \textit{LocationOf} \mid \textit{CuisineOf} \mid \cdots
\end{array}
$$

Semantics of FOL

- Model-theoretic approach:
  - FOL terms (objects): denote elements in a domain
  - Atomic formulas are:
    - If properties, sets of domain elements
    - If relations, sets of tuples of elements
- Formulas based on logical operators:

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
False	*False*	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*	*True*
True	*False*	*False*	*False*	*True*	*False*
True	*True*	*False*	*True*	*True*	*True*

Inference

- Standard AI-type logical inference procedures
  - Modus Ponens
  - Forward-chaining, Backward Chaining
  - Abduction
  - Resolution
  - Etc,..
- We'll assume we have a prover

Representing Events

- Initially, single predicate with some arguments
  - Serves(Maharani,IndianFood)

Representing Events

- Initially, single predicate with some arguments
  - Serves(Maharani,IndianFood)
  - Assume # ags = # elements in subcategorization frame

Representing Events

- Initially, single predicate with some arguments
  - Serves(Maharani,IndianFood)
  - Assume # ags = # elements in subcategorization frame
- Example:
  - 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.

Events

- Issues?

Events

- Issues?
  - Arity – how can we deal with different #s of arguments?

Events

- Issues?
  - Arity – how can we deal with different #s of arguments?
- One predicate per frame
  - $Eating_1$(Speaker)
  - $Eating_2$(Speaker,TS)
  - $Eating_3$(Speaker,TS,Desk)
  - $Eating_4$(Speaker,Desk)
  - $Eating_5$(Speaker,TS,Lunch)
  - $Eating_6$(Speaker,TS,Lunch,Desk)

Events (Cont'd)

- Good idea?

Events (Cont'd)

- Good idea?
  - Despite the names, actually unrelated predicates

Events (Cont'd)

- Good idea?
  - Despite the names, actually unrelated predicates
    - Can't derive obvious info
      - E.g. I ate a turkey sandwich for lunch at my desk
        - Entails all other sentences

Events (Cont'd)

- Good idea?
  - Despite the names, actually unrelated predicates
    - Can't derive obvious info
      - E.g. I ate a turkey sandwich for lunch at my desk
        - Entails all other sentences
    - Can't directly associate with other predicates

Events (Cont'd)

- Good idea?
  - Despite the names, actually unrelated predicates
    - Can't derive obvious info
      - E.g. I ate a turkey sandwich for lunch at my desk
        - Entails all other sentences
    - Can't directly associate with other predicates
- Could write rules to implement implications

Events (Cont’d)

- Good idea?
  - Despite the names, actually unrelated predicates
    - Can’t derive obvious info
      - E.g. I ate a turkey sandwich for lunch at my desk
        - Entails all other sentences
    - Can’t directly associate with other predicates
- Could write rules to implement implications
  - But?
    - Intractable in the large
      - Like the subcat problem generally.

Variabilizing

- Create predicate with maximum possible arguments
  - Include appropriate args
  - Maintains connections

$$\exists w,x,y\, Eating(Speaker, w, x, y)$$

$$\exists w,x\, Eating(Speaker, TS, w, x)$$

$$\exists w\, Eating(Speaker, TS, w, Desk)$$

$$Eating(Speaker, TS, Lunch, Desk)$$

Variabilizing

- Create predicate with maximum possible arguments
  - Include appropriate args
  - Maintains connections

$$\exists w, x, y\, Eating(Speaker, w, x, y)$$

$$\exists w, x\, Eating(Speaker, TS, w, x)$$

$$\exists w\, Eating(Speaker, TS, w, Desk)$$

$$Eating(Speaker, TS, Lunch, Desk)$$

- Better?

Variabilizing

- Create predicate with maximum possible arguments
  - Include appropriate args
  - Maintains connections

$$\exists w, x, y\, Eating(Speaker, w, x, y)$$

$$\exists w, x\, Eating(Speaker, TS, w, x)$$

$$\exists w\, Eating(Speaker, TS, w, Desk)$$

$$Eating(Speaker, TS, Lunch, Desk)$$

- Better?
  - Yes, but
    - Too many commitments – assume all details show up

Variabilizing

- Create predicate with maximum possible arguments
  - Include appropriate args
  - Maintains connections

$$\exists w, x, y\, Eating(Speaker, w, x, y)$$

$$\exists w, x\, Eating(Speaker, TS, w, x)$$

$$\exists w\, Eating(Speaker, TS, w, Desk)$$

$$Eating(Speaker, TS, Lunch, Desk)$$

- Better?
  - Yes, but
    - Too many commitments – assume all details show up
    - Can’t individuate – don’t know if same event

Events - Finalized

- Neo-Davidsonian representation:
  - Distill event to single argument for event itself
  - Everything else is additional predication

$$\exists e Eating(e) \wedge Eater(e, Speaker) \wedge Eaten(e, TS) \wedge Meal(e, Lunch) \wedge Location(e, Desk)$$

Events - Finalized

- Neo-Davidsonian representation:
  - Distill event to single argument for event itself
  - Everything else is additional predication

$$\exists e Eating(e) \wedge Eater(e, Speaker) \wedge Eaten(e, TS) \wedge Meal(e, Lunch) \wedge Location(e, Desk)$$

- Pros:

Events - Finalized

- Neo-Davidsonian representation:
  - Distill event to single argument for event itself
  - Everything else is additional predication

$$\exists e Eating(e) \wedge Eater(e, Speaker) \wedge Eaten(e, TS) \wedge Meal(e, Lunch) \wedge Location(e, Desk)$$

- Pros:
  - No fixed argument structure
    - Dynamically add predicates as necessary

Events - Finalized

- Neo-Davidsonian representation:
  - Distill event to single argument for event itself
  - Everything else is additional predication

$$\exists e\, Eating(e) \wedge Eater(e, Speaker) \wedge Eaten(e, TS) \wedge Meal(e, Lunch) \wedge Location(e, Desk)$$

- Pros:
  - No fixed argument structure
    - Dynamically add predicates as necessary
  - No extra roles

Events - Finalized

- Neo-Davidsonian representation:
  - Distill event to single argument for event itself
  - Everything else is additional predication

$$\exists e Eating(e) \wedge Eater(e, Speaker) \wedge Eaten(e, TS) \wedge Meal(e, Lunch) \wedge Location(e, Desk)$$

- Pros:
  - No fixed argument structure
    - Dynamically add predicates as necessary
  - No extra roles
  - Logical connections can be derived

Representing Time

- Temporal logic:
  - Includes tense logic to capture verb tense infor
- Basic notion:
  - Timeline:
    - From past to future
    - Events associated with points or intervals on line
      - Ordered by positioning on line
    - Current time
      - Relative order gives past/present/future

Temporal Information

- I arrived in New York.
- I am arriving in New York.
- I will arrive in New York.
  - Same event, differ only in tense

$$\exists e Arriving(e) \wedge Arriver(e, Speaker) \wedge Destination(e, NY)$$

- Create temporal representation based on verb tense
  - Add predication about event variable
  - Temporal variables represent:
    - Interval of event
    - End point of event
    - Predicates link end point to current time

Temporal Representation

$$\exists e,i,n,t Arriving(e) \wedge Arriver(e, Spea\text{ker}) \wedge Destination(e, NY)$$
$$\wedge IntervalOf(e,i) \wedge EndPo\text{int}(i,e) \wedge \text{Pr}\,ecedes(e, Now)$$

$$\exists e,i,n,t Arriving(e) \wedge Arriver(e, Spea\text{ker}) \wedge Destination(e, NY)$$
$$\wedge IntervalOf(e,i) \wedge MemberOf(i, Now)$$

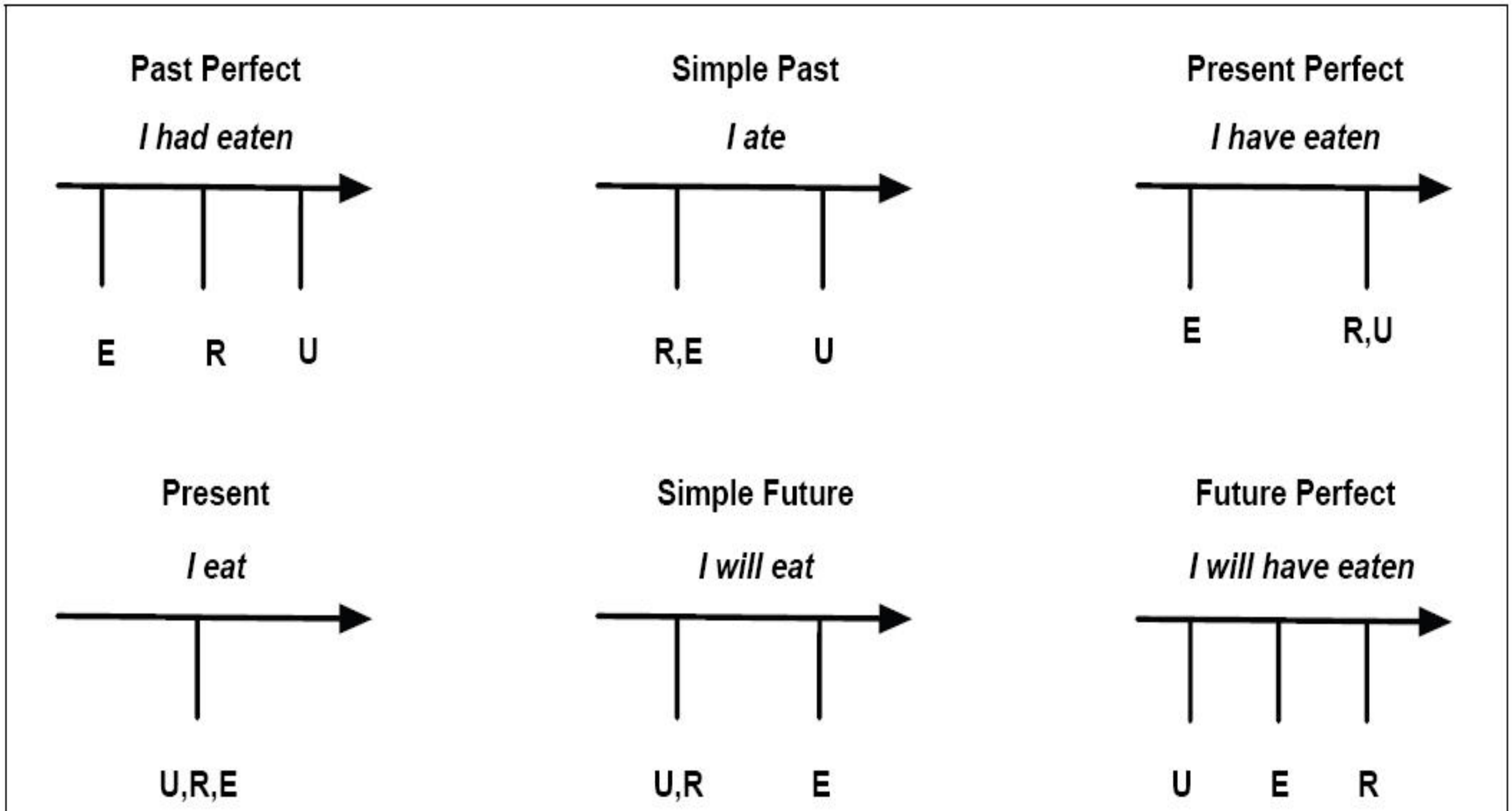
$$\exists e,i,n,t Arriving(e) \wedge Arriver(e, Spea\text{ker}) \wedge Destination(e, NY)$$
$$\wedge IntervalOf(e,i) \wedge EndPo\text{int}(i,n) \wedge \text{Pr}\,ecedes(Now, e)$$

More Temp Rep

- Flight 902 arrived late.
- Flight 902 had arrived late.

- Does the current model cover this?
  - Not really
  - Need additional notion:
    - Reference point
      - As well as current time, event time
        - Current model: current = utterance time = reference point

Reichenbach's Tense Model



Meaning Representation for Computational Semantics

- Requirements:
  - Verifiability, Unambiguous representation, Canonical Form, Inference, Variables, Expressiveness
- Solution:
  - First-Order Logic
    - Structure
    - Semantics
    - Event Representation
- Next: Semantic Analysis
  - Deriving a meaning representation for an input

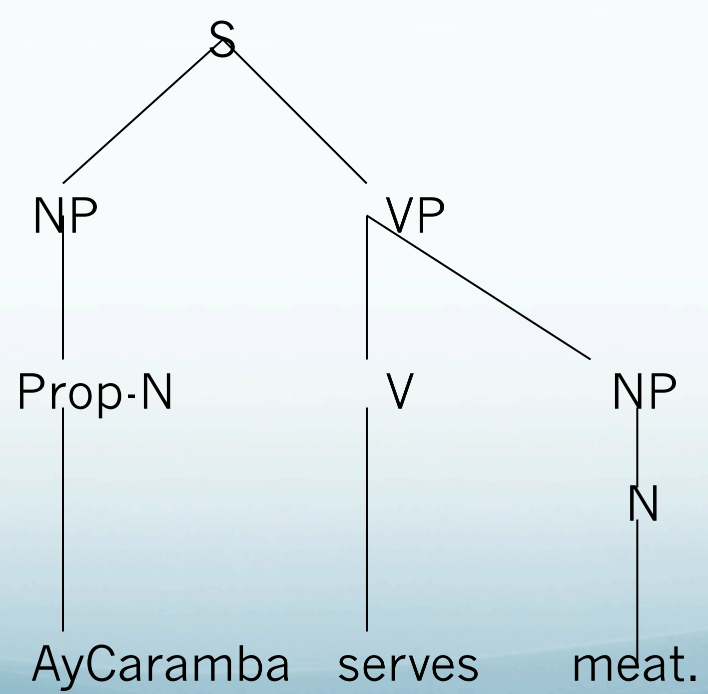
Syntax-driven Semantic Analysis

- Key: Principle of Compositionality
  - Meaning of sentence from meanings of parts
    - E.g. groupings and relations from syntax
- Question: Integration?
- Solution 1: Pipeline
  - Feed parse tree and sentence to semantic unit
  - Sub-Q: Ambiguity:
    - Approach: Keep all analyses, later stages will select

Simple Example

- AyCaramba serves meat.

$$\exists e\ Isa(e, Serving) \wedge Server(e, AyCaramba) \wedge Served(e, Meat)$$



Rule-to-Rule

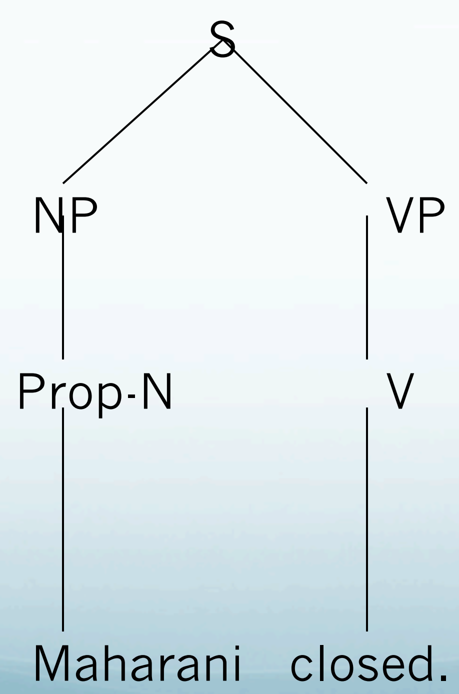
- Issue:
  - How do we know which pieces of the semantics link to what part of the analysis?
  - Need detailed information about sentence, parse tree
    - Infinitely many sentences & parse trees
    - Semantic mapping function per parse tree => intractable
- Solution:
  - Tie semantics to finite components of grammar
    - E.g. rules & lexicon
  - Augment grammar rules with semantic info
    - Aka "attachments"
      - Specify how RHS elements compose to LHS

Semantic Attachments

- Basic structure:
  - A-> a1....an {f(aj.sem,...ak.sem)}
  - A.sem
- Language for semantic attachments
  - Arbitrary programming language fragments?
    - Arbitrary power but hard to map to logical form
    - No obvious relation between syntactic, semantic elements
  - Lambda calculus
    - Extends First Order Predicate Calculus (FOPC) with function application
  - Feature-based model + unification
- Focus on lambda calculus approach

Basic example

- Input: Maharani closed.
- Target output: Closed(Maharani)



Semantic Analysis Example

- Semantic attachments:
  - Each CFG production gets semantic attachment
- Maharani
  - ProperNoun -> Maharani {Maharani}
    - FOL constant to refer to object
  - NP -> ProperNoun {ProperNoun.sem}
    - No additional semantic info added

Semantic Attachment Example

- Phrase semantics is function of SA of children

- More complex functions are parameterized
  - E.g. Verb -> closed { λ x.Closed(x) }
    - Unary predicate:
      - 1 arg = subject, not yet specified
  - VP -> Verb {Verb.sem}
    - No added information
  - S -> NP VP {VP.sem(NP.sem)}
    - Application= λ x.Closed(x)(Maharanii) = Closed(Maharani)

Semantic Attachment

- General pattern:
  - Grammar rules mostly lambda reductions
    - Functor and arguments

  - Most representation resides in lexicon

Refining Representation

- Add
  - Neo-Davidsonian event-style model
  - Complex quantification
- Example II
  - Input: Every restaurant closed.
  - Target:

$$\forall x\, Restaurant(x) \Rightarrow \exists e\, Closed(e) \wedge ClosedThing(e,x)$$

Refining Representation

- Idea: $\forall x \, \mathrm{Re}\, staurant(x)$
  - Good enough?
    - No: roughly 'everything is a restaurant'
    - Saying something about all restaurants – nuclear scope
- Solution: Dummy predicate
  $\forall x \, \mathrm{Re}\, staurant(x) \Rightarrow Q(x)$
  - Good enough?
    - No: no way to get Q(x) from elsewhere in sentence
- Solution: Lambda

  $\lambda Q.\forall x \, \mathrm{Re}\, staurant(x) \Rightarrow Q(x)$

Updating Attachments

- Noun -> restaurant $\{\lambda x.Restaurant(x)\}$
- Nominal -> Noun { Noun.sem }
- Det -> Every $\{ \lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x) \}$
- NP -> Det Nominal { Det.sem(Nom.sem) }

$$\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)(\lambda x.\mathrm{Re}\,staurant(x))$$

$$\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)(\lambda y.\mathrm{Re}\,staurant(y))$$

$$\lambda Q.\forall x \lambda y.\mathrm{Re}\,staurant(y)(x) \Rightarrow Q(x)$$

$$\lambda Q.\forall x\,\mathrm{Re}\,staurant(x) \Rightarrow Q(x)$$

Full Representation

- Verb -> close $\{\lambda x.\exists e Closed(e) \wedge ClosedThing(e,x)\}$
- VP -> Verb { Verb.sem }
- S -> NP VP { NP.sem(VP.sem) }

$$\lambda Q.\forall x Restaurant(x) \Rightarrow Q(x)(\lambda y.\exists e Closed(e) \wedge ClosedThing(e,y))$$

$$\forall x Restaurant(x) \Rightarrow \lambda y.\exists e Closed(e) \wedge ClosedThing(e,y)(x)$$

$$\forall x Restaurant(x) \Rightarrow \exists e Closed(e) \wedge ClosedThing(e,x)$$

Generalizing Attachments

  - ProperNoun -> Maharani {Maharani}
- Does this work in the new style?
  - No, we turned the NP/VP application around
- New style: $\lambda x.x(\text{Maharani})$

More

- Determiner
- Det -> a { $\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)$ }
- a restaurant $\lambda Q.\exists x \mathrm{Re}\,staurant(x) \wedge Q(x)$

- Transitive verb:
  - VP -> Verb NP { Verb.sem(NP.sem) }
  - Verb -> opened

$$\lambda w.\lambda z.w(\lambda x.\exists e Opened(e) \wedge Opener(e,z) \wedge OpenedThing(e,w))$$

Strategy for Semantic Attachments

- General approach:
  - Create complex, lambda expressions with lexical items
    - Introduce quantifiers, predicates, terms
  - Percolate up semantics from child if non-branching
  - Apply semantics of one child to other through lambda
    - Combine elements, but don't introduce new

Sample Attachments

Grammar Rule	Semantic Attachment
$S \rightarrow NP\ VP$	$\{NP.sem(VP.sem)\}$
$NP \rightarrow Det\ Nominal$	$\{Det.sem(Nominal.sem)\}$
$NP \rightarrow ProperNoun$	$\{ProperNoun.sem\}$
$Nominal \rightarrow Noun$	$\{Noun.sem\}$
$VP \rightarrow Verb$	$\{Verb.sem\}$
$VP \rightarrow Verb\ NP$	$\{Verb.sem(NP.sem)\}$
$Det \rightarrow every$	$\{\lambda P.\lambda Q.\forall x P(x) \Rightarrow Q(x)\}$
$Det \rightarrow a$	$\{\lambda P.\lambda Q.\exists x P(x) \wedge Q(x)\}$
$Noun \rightarrow restaurant$	$\{\lambda r.Restaurant(r)\}$
$ProperNoun \rightarrow Matthew$	$\{\lambda m.m(Matthew)\}$
$ProperNoun \rightarrow Franco$	$\{\lambda f.f(Franco)\}$
$ProperNoun \rightarrow Frasca$	$\{\lambda f.f(Frasca)\}$
$Verb \rightarrow closed$	$\{\lambda x.\exists e Closed(e) \wedge ClosedThing(e,x)\}$
$Verb \rightarrow opened$	$\{\lambda w.\lambda z.w(\lambda x.\exists e Opened(e) \wedge Opener(e,z) \wedge Opened(e,x))\}$

Quantifier Scope

- Ambiguity:
  - *Every restaurant has a menu*

$$\forall x\, Restaurant(x) \Rightarrow \exists y (Menu(y) \wedge (\exists e Having(e) \wedge Haver(e,x) \wedge Had(e,y)))$$

  - Readings:
    - all have a menu;
    - all have same menu
  - Only derived one

$$\exists y Menu(y) \wedge \forall x (Restaurant(x) \Rightarrow \exists e Having(e) \wedge Haver(e,x) \wedge Had(e,y)))$$

  - Potentially O(n!) scopings (n=# quantifiers)

- There are approaches to describe ambiguity efficiently and recover all alternatives.

Earley Parsing with Semantics

- Implement semantic analysis
  - In parallel with syntactic parsing
    - Enabled by compositional approach
- Required modifications
  - Augment grammar rules with semantic field
  - Augment chart states with meaning expression
  - Completer computes semantics – e.g. unifies
    - Can also fail to unify
      - Blocks semantically invalid parses
    - Can impose extra work

Sidelight: Idioms

- Not purely compositional
  - E.g. kick the bucket = die
  - tip of the iceberg = beginning
- Handling:
  - Mix lexical items with constituents (word nps)
  - Create idiom-specific const. for productivity
  - Allow non-compositional semantic attachments
- Extremely complex: e.g. metaphor