

CS 6890: Lecture 6

Vladimir Kulyukin

Department of Computer Science

Utah State University

Outline

- Semantic Analysis

Meaning Representation

- Fundamental assumption: *Meaning can be captured with formal representations.*
- A meaning representation is a bridge between linguistic and non-linguistic knowledge (common sense reasoning).
- Examples of tasks that require non-linguistic knowledge:
 - following a recipe;
 - writing an essay;
 - learning to use a software package by reading a manual.

Semantic Analysis

Semantic analysis is a process through which a meaning representation is assigned to an input.

Requirements for Knowledge Representation

- Verifiability
- Non-ambiguity
- Canonicity
- Inferential Power (Support for Inference)
- Expressiveness

Verifiability

- KR must enable the system to compare the state of affairs described by a representation with the state of affairs as modeled by a knowledge database.
- If there is a match between a representation and the knowledge base, the system is said to have verified the representation.

Example

- *Does Maharani serve vegetarian food?*
- Suppose that we somehow represent the meaning of this statement as
- *Serves(Maharani, VegetarianFood).*
- Suppose that we have a knowledge database of similar propositions.
- At this point, we either find a match in the database or, if we do not find a match and our database is not complete, we say that we do not know.

Lack of Ambiguity

- Knowledge representations should not be ambiguous.
- No matter how ambiguous the raw input is, the resulting knowledge representation constructed from it should have as few ambiguities as possible.

Example

- *I want to eat some place close to my work.*
- This sentence can be interpreted as the speaker wanting to eat at some place that serves food and close to work.
- This sentence can also be interpreted as the speaker wanting to eat some place that is located close to his/her work.

Ambiguity and Vagueness

- Ambiguity should not be confused with vagueness.
- An ambiguous statement has a number of different interpretations.
- A vague statement has one interpretation some elements of which may be unspecified.
- Example: *I want to eat Italian food.*
- This sentence is vague but unambiguous.

Canonicity

- Consider the following sentences:
 - Does Maharani have vegetarian dishes?
 - Do they have vegetarian food at Maharani?
 - Are vegetarian dishes served at Maharani?

Canonicity

- The previous sentences have very different morphological and syntactic structures.
- They have the same meaning.
- There are two choices for meaning representation:
 - Store a separate meaning representation for each syntactic sentence.
 - Have one meaning representation (canonical form) for syntactically different structures that mean the same thing.

Canonicity: Pros and Cons

- Pros
 - The design and maintenance of our knowledge database is simplified.
 - The implementation of inference algorithms is simplified.
- Cons
 - The sentence analysis becomes more complex, because we have to extract a canonical representation from a great variety of syntactic structures.

Canonicity

- Our general approach will be as follows: Identify systemic relationships among word senses and grammatical constructions that can be used to make computing canonical representations tractable.
- Our knowledge of syntactic structures should enable us to decide that these sentences mean the same thing:
 - Maharani serves vegetarian dishes.
 - Vegetarian dishes are served by Maharani.

Support for Inference

- Canonicity has limits: Suppose that we can compute canonical structures, we can answer yes/no questions by simply matching the canonical representation with the representations in the database.
- Suppose that we have an input question *Can vegetarians eat at Maharani?*
- Suppose our database has a representation stating that Maharani serves vegetarian food.
- Canonicity is insufficient to answer the input question. We need inference:
 - vegetarian restaurants serve vegetarian dishes;
 - Maharani serves vegetarian dishes;
 - hence, Maharani is a vegetarian restaurant;
 - vegetarians eat at vegetarian restaurants;
 - hence, vegetarians can eat at Maharani.

Support for Inference

- Suppose that the input sentence is *I'd like to find a restaurant where I can get vegetarian food.*
- The user wants to know all unknown entities that are 1) restaurants and 2) serve vegetarian food.
- Operationally speaking, we have to translate the input question into something like `Serves(x, VegetarianFood)` and retrieve every entity that matches `x`.

Expressive Power

- Useful meaning representation languages must express a sufficient variety of meanings.
- The ideal standard: have a common lingua franca of semantic representations that can express the meaning of any linguistic input.

Meaning Structure of Language

- Human languages have a form of predicate-argument structure: who did what how and when.
- Human languages are semantically compositional: they allow the creation of complex semantic structures from more simple (primitive) semantic structures.
- The above observations translate into two important requirements for a meaning representation language: 1) it must capture the predicate-argument structure and 2) it must allow compositionality.

Predicate-Argument Structure

- Consider these sentences:
 1. I want Italian food.
 2. I want to spend less than five dollars.
 3. I want it to be close by here.
- These sentences have the following syntactic argument frames:
 1. NP want NP
 2. NP want Inf-NP
 3. NP want NP Inf-NP

Predicate-Argument Structure

- I want Italian food.
- NP want NP.
- The predicate is want
 - The predicate has 2 arguments.
 - Both arguments are syntactically realized as NPs.
 - The first argument is pre-verbal and plays the role of the subject.
 - The second argument is post-verbal and plays the role of the direct object.

Predicate-Argument Structure

- The previous analysis covers the syntax.
- What about semantics? We can observe that in addition to syntactic positions within a sentence, we have semantic roles and semantic restrictions that can be placed on these roles.
- Example: In the syntactic structure *NP want NP* it is the pre-verbal NP that does the wanting and the post-verbal NP specifies what the pre-verbal NP wants.

First-Order Predicate Calculus

- FOPC is a flexible, well-understood, and computationally tractable (in many cases).
- FOPC satisfies the verifiability, inference, and expressiveness requirements.
- FOPC is attractive, because it makes very few requirements on how things ought to be represented.
- In FOPC, the represented world (universe of discourse) consists of objects, object properties, and relations among objects.

A (Very Brief) Intro to FOPC

Term

- A Term in FOPC means to represent objects.
- There are 3 types of terms in FOPC:
 - Constant
 - Function
 - Variable

Constants

- Constants are references to specific objects that exist in the represented world.
- Each FOPC constant refers to exactly one object.
- Examples: A, B, Maharani, USU, OldMain.

Functions

- Functions are references to objects that are often expressed in English as genitives: the location of Maharani, Maharani's location, the father of Arthur, Arthur's father, etc.
- FOPC translations:
 - LocationOf(Maharani)
 - FatherOf(Arthur)

Functions

- Functions have the same appearance as single argument predicates.
- Functions refer to unique objects; they do not evaluate, like predicates, to true or false.
- LocationOf(Maharani) refers to the physical location of a restaurant. It does not evaluate to true or false.
- Functions refer to unique objects without having to associate a name with them.

Variables

- Variables (typically lower-case letters) refer to objects without making a reference to any specific object.
- Variables refer to to a particular unknown objects or about all objects in some represented world (universe of discourse).

Predicates

- A predicate refers to a relation that holds among terms (objects) in a given universe of discourse.
- Examples:
 - Maharani serves vegetarian food.
 - Serves(Maharani, VegetarianFood).
 - Maharani is a restaurant.
 - Restaurant(Maharani).

Logical Connectives

- Logical connectives of FOPC are a means to satisfy the compositionality requirement, i.e. to compose more complex semantic structures out of less complex semantic structures.
- Example:
 - I only have five dollars and I don't have a lot of time.

$\text{Have}(\text{Speaker}, \text{FiveDollars}) \wedge \neg \text{Have}(\text{Speaker}, \text{LotOfTime})$

FOPC Semantics

Old Main is near Eccles.

The constants : OldMain, Eccles.

The functions : LocationOf(OldMain), LocationOf(Eccles).

The predicate : Near

The result : Near(LocationOf(OldMain), LocationOf(Eccles)).

Variables and Quantifiers

- Variables are used in two ways: to refer to a given unknown object or to refer to all objects in a collection.
- These two uses are made possible through quantifiers \exists , \forall
- The need for existential quantifiers arises when there is an indefinite noun phrase in English.

Example

A restaurant that serves Italian food near the dorm.

$(\exists x)[\text{Restaurant}(x) \wedge \text{Serves}(x, \text{MexicanFood}) \wedge \text{Near}(\text{LocationOf}(x), \text{LocationOf}(\text{Dorm}))]$

Example

All vegetarian restaurants serve vegetarian food.

$$(\forall x)[\text{VegetarianRestaurant}(x) \Rightarrow \text{Serves}(x, \text{VegetarianFood})]$$

Inference

Modus ponens

α

$\alpha \Rightarrow \beta$

β

Example

VegetarianRestaurant(Rudys)

$(\forall x)[\text{VegetarianRestaurant}(x) \Rightarrow \text{Serves}(x, \text{VegetarianFood})]$

$\text{Serves}(\text{Rudys}, \text{VegetarianFood})$

Modus Ponens Implementation

- Forward chaining
 - as propositions are added to the database, all applicable rules are run to completion.
- Backward chaining
 - try to prove the proposition by using the facts and rules of the database.

Concept Representation

- Categories
- Events

Categories

- A category is a set of elements that share a set of features.
- Unary predicates are the most common way of representing categories.
- Every member of a category set will have a set of unary predicates that are true of it.
- Example:
VegetarianRestaurant(Maharani).

Categories

- If we use predicates to represent categories, we effectively turn categories into relations.
- One disadvantage of doing so is that it becomes difficult to make assertions about categories themselves.
- Example: MostPopular(Maharani, VegetarianRestaurant) is not a legal FOPC statement if VegetarianRestaurant is a predicate.

Categories

- An alternative representation methodology is to represent all categories as objects.
- This technique is known as reification.
- Reification is achieved with two relations: ISA and AKO (a-kind-of).
- ISA holds between objects and categories.
 - ISA(Maharani, VegetarianRestaurant).
- AKO holds between two categories.
 - AKO(VegetarianRestaurant, Restaurant).

Events

- Consider the following sentences:
 1. I ate.
 2. I ate a turkey sandwich.
 3. I ate a turkey sandwich at my desk.
 4. I ate at my desk.
 5. I ate lunch.
 6. I ate a turkey sandwich for lunch.
 7. I ate a turkey sandwich for lunch at my desk.

Events

- The first solution is to map each syntactic pattern onto a corresponding predicate:
 1. Eating₁(Speaker)
 2. Eating₂(Speaker, TurkeySandwich)
 3. Eating₃(Speaker, TurkeySandwich, Desk)
 4. Eating₄(Speaker, Desk)
 5. Eating₅(Speaker, Lunch)
 6. Eating₆(Speaker, TurkeySandwich, Lunch)
 7. Eating₇(Speaker, TurkeySandwich, Lunch, Desk)

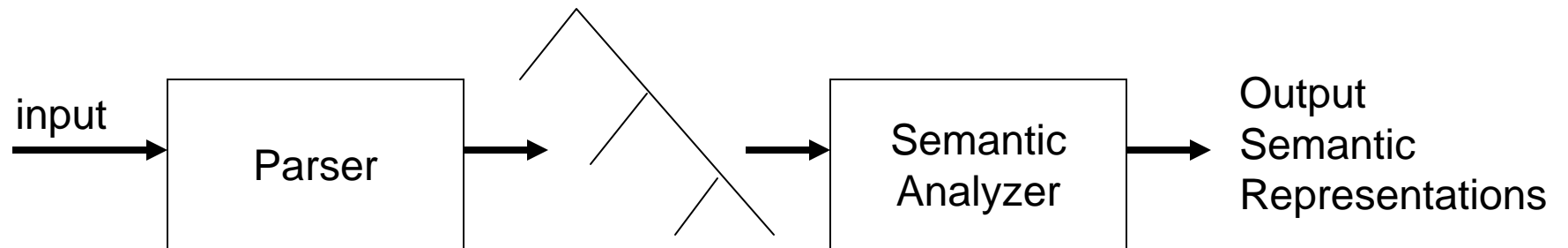
Events

- This approach captures the different syntactic patterns but there is nothing that ties these logical statements together.
- For example,
 - if $\text{Eating}_7(\text{Speaker}, \text{TurkeySandwich}, \text{Lunch}, \text{Desk})$ is true,
 - then $\text{Eating}_1(\text{Speaker})$, $\text{Eating}_2(\text{Speaker}, \text{TurkeySandwich})$, $\text{Eating}_3(\text{Speaker}, \text{TurkeySandwich}, \text{Desk})$ are also true.

Syntax-Driven Semantic Analysis

- Syntax-driven semantic analysis is to assign meaning representations to inputs based on static knowledge from the lexicon and the grammar.
- Syntax-driven semantic analysis is based on the principle of compositionality: the meaning of a sentence can be composed from the meaning of its parts.

Syntax-Driven Semantic Analysis



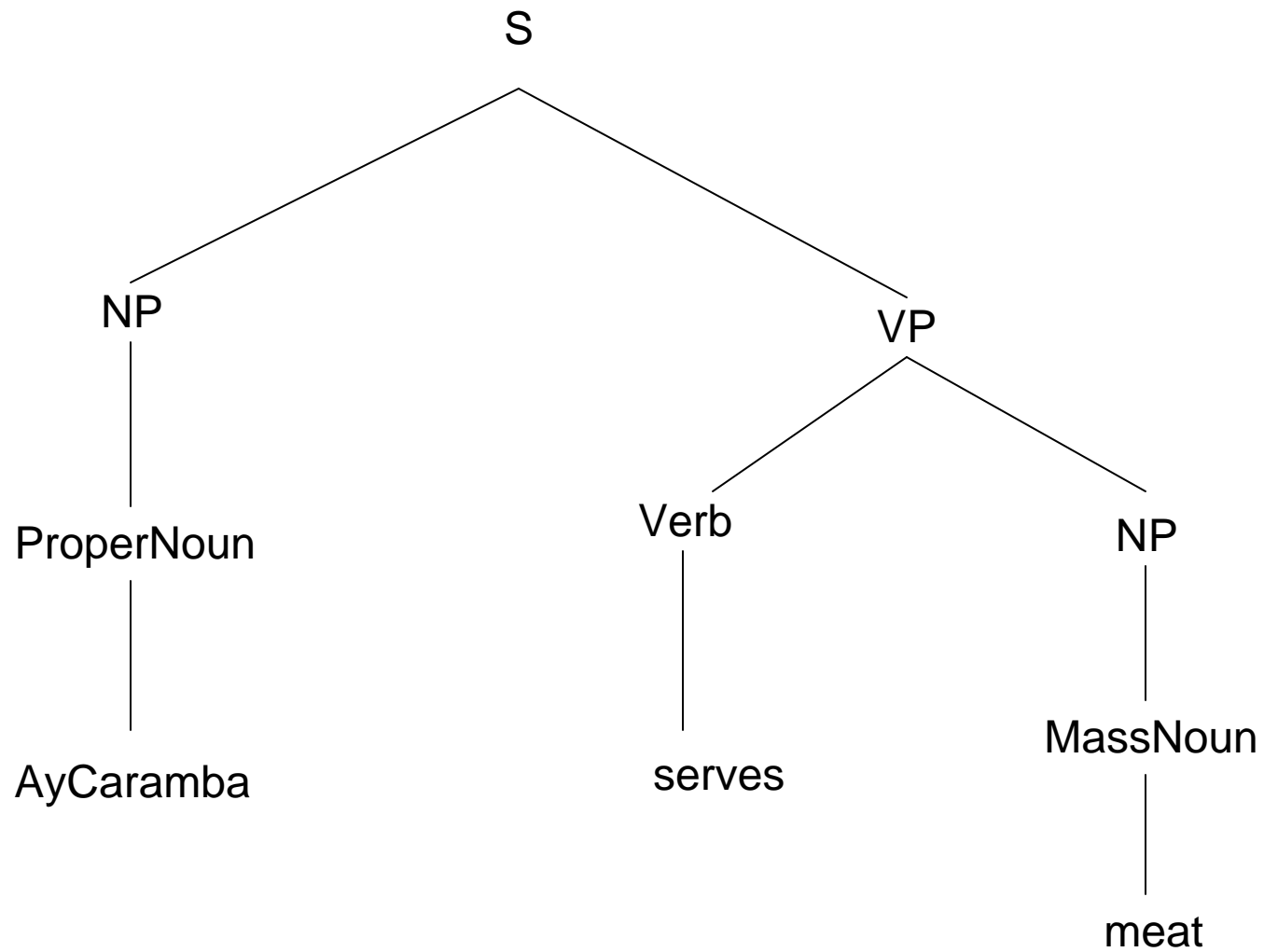
Syntax-Driven Semantic Analysis

- Ambiguities arising from the syntax and the lexicon may lead to multiple representations
- The job of selecting a representation is not the job of the semantic analyzer.

Example

Ay Caramba serves meat.

Example



Example

$$\exists e \left[\begin{array}{l} \textit{ISA}(e, \textit{Serving}) \wedge \textit{Server}(e, \textit{AyCaramba}) \wedge \\ \textit{Served}(e, \textit{Meat}) \end{array} \right]$$

Two Questions

- What does it mean for syntactic components to have meanings?
- How can these meanings be composed into larger meanings?

Semantic Augmentation to CFG Rules

Let A be a syntactic construct. Let $A.sem$ be the meaning representation of A . The format of the semantically augmented CFG rule is as follows :

$$A \rightarrow \alpha_1 \dots \alpha_n \quad \{f(\alpha_j.sem, \dots, \alpha_k.sem)\}, 1 \leq j \leq n, 1 \leq k \leq n.$$

Lexical Items as FOPC Constants

- Lexical items can be represented as FOPC constants.
- Here are the rules:
 - ProperNoun → AyCaramba {AyCaramba}
 - MassNoun → meat {Meat}

Meaning Composition

- The upper NPs obtain their meaning representations from the meanings of their children.
- Examples:
 - NP → ProperNoun {ProperNoun.sem}
 - NP → MassNoun {MassNoun.sem}

Verb Meaning Representation

- Verbs are associated with events.
- For example, the verb serve can be associated with the Serving event that involves the Server and something Served.

Verb Meaning Representation

$$\textit{Verb} \rightarrow \textit{serves} \left\{ \begin{array}{l} \exists e, x, y \textit{ ISA}(e, \textit{Serving}) \wedge \textit{Server}(e, x) \wedge \\ \textit{Served}(e, y) \end{array} \right\}$$

Verb Meaning Representation

- VP in the syntactic tree dominates two lexical items: *serves* and *meat*.
- We cannot copy the meaning of Verb.sem and NP.sem into the VP.sem.
- We have to come up with a method that replaces the variable *y* with the FOPC constant *Meat*.

Verb Meaning Representation

$$\exists e, x \left[\begin{array}{l} \textit{ISA}(e, \textit{Serving}) \wedge \textit{Server}(e, x) \wedge \\ \textit{Served}(e, \textit{Meat}) \end{array} \right]$$

Verb Meaning Representation

- The VP semantic attachments must have two capabilities:
 - The means to know which variables in the Verb.sem must be replaced;
 - The means to carry out such a replacement.

Lambda Calculus

Church, A. (1940). A formulation of a simple theory of types. *Journal of Symbolic Logic*, 5: 56-68.

Lambda Calculus

$\lambda x P(x)$.

$\lambda x P(x)(A) = P(A)$.

Examples:

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

$\lambda x \lambda y \text{ Near}(x, y)(\text{Maharani}) = \lambda y \text{ Near}(\text{Maharani}, y)$.

$\lambda y \text{ Near}(\text{Maharani}, y)(\text{AyCaramba}) =$

$\text{Near}(\text{Maharani}, \text{AyCaramba})$.

Verb Meaning Representation

Verb → *serves*

$\{\lambda x \lambda y \exists e, y \text{ ISA}(e, \text{Serving}) \wedge \text{Server}(e, y) \wedge \text{Served}(e, x)\}$

VP → *Verb NP* {*Verb.sem*(*NP.sem*)}

S → *NP VP* {*VP.sem*(*NP.sem*)}

Quantification

A restaurant serves meat.

Problem with Quantification and Lambda Reduction

$\exists x \text{ ISA}(x, \text{Restaurant}).$

$\exists e, x \left[\begin{array}{l} \text{ISA}(e, \text{Serving}) \wedge \text{Server}(e, x) \wedge \text{Served}(e, \text{Meat}) \\ \wedge \text{ISA}(x, \text{Restaurant}) \end{array} \right]$

$\exists e \left[\begin{array}{l} \text{ISA}(e, \text{Serving}) \wedge \text{Server}(e, \exists x \text{ ISA}(x, \text{Restaurant})) \\ \wedge \text{Served}(e, \text{Meat}) \end{array} \right]$

Complex Terms

- The FOPC syntax must be extended to include situations when quantified expressions can be used in place of terms.
- This is the purpose of complex-terms.
- A complex term has the following structure: $\langle \text{Quantifier variable body} \rangle$.

Complex Terms

$$\exists e \left[\begin{array}{l} \textit{ISA}(e, \textit{Serving}) \wedge \\ \textit{Server}(e, \langle \exists x \textit{ISA}(x, \textit{Restaurant}) \rangle) \wedge \\ \textit{Served}(e, \textit{Meat}) \end{array} \right]$$

Complex Term Reduction

$P(\langle\langle \text{Quantifier variable body} \rangle\rangle) \Rightarrow$

Quantifier variable body Connective $P(\text{variable})$.

When Quantifier is \exists , Connective is \wedge .

When Quantifier is \forall , Connective is \Rightarrow .

Complex Term Reduction: Example

$Server(e, \langle \exists x ISA(x, Restaurant) \rangle)$

\Rightarrow

$\exists x ISA(x, Restaurant) \wedge Server(e, x).$

Complex Term Component Access

- Let A be a complex term.
- $A.sem.quantifier$ accesses the quantifier.
- $A.sem.variable$ accesses the variable.
- $A.sem.body$ accesses the body.

Back to the Example

Noun \rightarrow restaurant {Restaurant}

Nominal \rightarrow Noun $\{\lambda x \text{ ISA}(x, \text{Noun.sem})\}$

Det \rightarrow a $\{\exists\}$

NP \rightarrow Det Nominal $\{\langle \text{Det.sem } x \text{ Nominal.sem}(x) \rangle\}$

Five Techniques of Semantic Representation

- The association of normal FOPC expressions with lexical items.
- The association of lambda expressions with lexical items.
- The copying of semantic values from children to parents.
- The computation of semantic representations through lambda reduction.
- The use of complex terms to allow quantified expressions to be treated as terms.

Integrating Semantic Analysis into the Early Parser

- The rules of the grammar are given a new field to contain their semantic attachments.
- The states in the chart are given a new field to hold the meaning representations of the constituent.
- The ENQUEUE function is altered so that when a complete state is entered into the chart its semantics are computed and stored in the state's semantic field.

Integrating Semantic Analysis into the Early Parser

```
procedure Enqueue(state, chart-entry) {  
    if ( isIncomplete(state) ) {  
        if ( state is not already in chart-entry ) {  
            Push(state, chart-entry);  
        }  
    }  
    else if ( ApplySemantics(state) succeeds ) {  
        if ( state is not already in chart-entry ) {  
            Push(state, chart-entry);  
        }  
    }  
}
```

```
procedure ApplySemantics(state) {  
    meaning-rep = Apply(state.semantic-attachment, state);  
    if ( meaning-rep != failure ) {  
        state.meaning-rep = meaning-rep;  
    }  
}
```