

CSE842: Natural Language Processing

Lecture 12: Feature Structures and Unification

2/23/2009

CSE842, Spring 2009, MSU

1

Limits of CFGs

- Recall that there are several things CFGs don't handle elegantly:
 - Agreement (A cat sleeps. Cats sleep.)
 $S \rightarrow NP VP$
 $NP \rightarrow Det Nom$
But these rules overgenerate, allowing, e.g., *they sleeps, *these dog
 - Subcategorization (*they find)

2/23/2009

CSE842, Spring 2009, MSU

2

$VP \rightarrow V$

$VP \rightarrow V NP$

But these also allow *She disappeared the elephant.

- We need to **constrain** the grammar rules to enforce e.g. number agreement and subcategorization differences

CFG Solution

- Encode constraints into the non-terminals
 - Noun/verb agreement
 $S \rightarrow SgS$
 $S \rightarrow PlS$
 $SgS \rightarrow SgNP SgVP$
 $SgNP \rightarrow SgDet SgNom$
 - Verb subcat:
 $IntransVP \rightarrow IntransV$
 $TransVP \rightarrow TransV NP$
- But this means huge proliferation of rules...

2/23/2009

CSE842, Spring 2009, MSU

3

2/23/2009

CSE842, Spring 2009, MSU

4

An Alternative

- View terminals and non-terminals as complex objects with associated **features**, which take on different values
- Write grammar rules whose application is constrained by tests on these features, e.g. $S \rightarrow NP VP$ (only if the NP and VP agree in number)
- We'll do this with **feature structures** and the constraint-based **unification** formalism

Feature Structures

- Sets of **feature-value pairs** where:
 - Features are atomic symbols
 - Values are atomic symbols or feature structures
 - Illustrated by **attribute-value matrix**

$$\begin{bmatrix} \textit{Feature}_1 & \textit{Value}_1 \\ \textit{Feature}_2 & \textit{Value}_2 \\ \dots & \dots \\ \textit{Feature}_n & \textit{Value}_n \end{bmatrix}$$

- Number feature

$$\begin{bmatrix} \textit{Num} & \textit{SG} \end{bmatrix}$$

- Number-person features

$$\begin{bmatrix} \textit{Num} & \textit{SG} \\ \textit{Pers} & 3 \end{bmatrix}$$

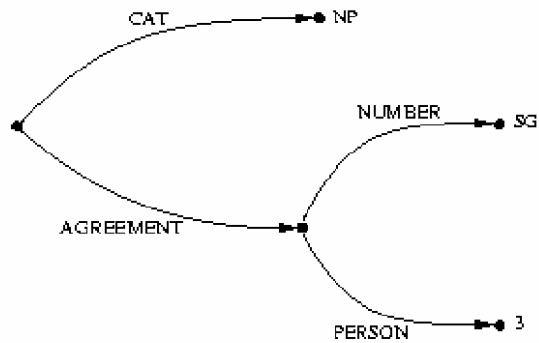
- Number-person-category features (3sgNP)

$$\begin{bmatrix} \textit{Cat} & \textit{NP} \\ \textit{Num} & \textit{SG} \\ \textit{Pers} & 3 \end{bmatrix}$$

- Feature values can be feature structures themselves: bundles of features
 - Useful when certain features commonly co-occur, e.g. number and person

$$\begin{bmatrix} \textit{Cat} & \textit{NP} \\ \textit{Agr} & \begin{bmatrix} \textit{Num} & \textit{SG} \\ \textit{Pers} & 3 \end{bmatrix} \end{bmatrix}$$

Graphical Notation for Feature Structures



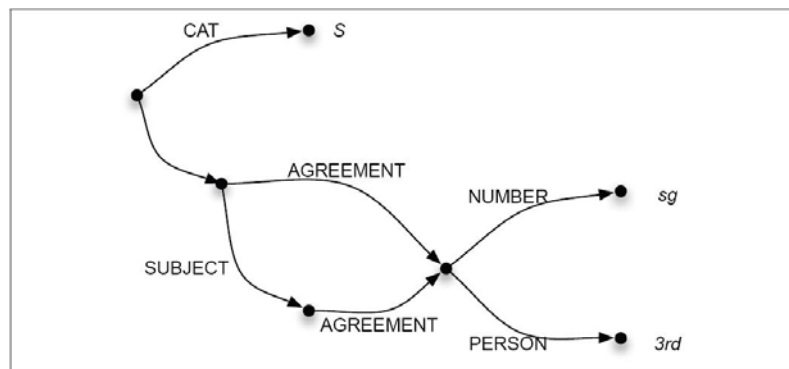
Reentrant Structures

- Feature structures may also contain features that share some feature structure as a value

$$\left[\begin{array}{l} \text{Cat } S \\ \text{Head} \left[\begin{array}{l} \text{Agr } 1 \left[\begin{array}{l} \text{Num } SG \\ \text{Pers } 3 \end{array} \right] \\ \text{Subj} \left[\text{Agr } 1 \right] \end{array} \right] \end{array} \right]$$

- Numerical indices indicate the shared values
- Path: <HEAD AGREEMENT NUMBER>

Reentrant Structures



Operations on Feature Structures

- What will we need to do to these structures?
 - Check the **compatibility** of two structures
 - **Merge** the information in two structures
- We can do both using **unification**

- We say that two feature structures **can be unified** if the component features that make them up are **compatible**

- $[\text{Num SG}] \cup [\text{Num SG}] = [\text{Num SG}]$
- $[\text{Num SG}] \cup [\text{Num PL}]$ fails!
- $[\text{Num SG}] \cup [\text{Num []}] = [\text{Num SG}]$
- $[\text{Num SG}] \cup [\text{Pers 3}] = \begin{bmatrix} \text{Num SG} \\ \text{Pers 3} \end{bmatrix}$

- Structures are compatible if they contain no features that are **incompatible**

- Unification of two feature structures:
 - Merger of the original structures into one larger structure
 - Are the structures compatible?
 - If so, return the union of all feature/value pairs
- Does the following unification succeed?

$$\begin{bmatrix} \text{Agr 1} & \begin{bmatrix} \text{Num SG} \\ \text{Pers 3} \end{bmatrix} \\ \text{Subj} & \begin{bmatrix} \text{Agr} & 1 \end{bmatrix} \end{bmatrix} \cup \begin{bmatrix} \text{Agr} & \begin{bmatrix} \text{Num SG} \\ \text{Pers 3} \end{bmatrix} \\ \text{Subj} & \begin{bmatrix} \text{Agr} & \begin{bmatrix} \text{Num PL} \\ \text{Pers 3} \end{bmatrix} \end{bmatrix}$$

Feature Structures in the Grammar

- Used to express syntactic constraints that would be difficult by using CFG alone.
- How do we incorporate feature structures into our grammars?
 - Assume that constituents are objects which have feature-structures associated with them
 - Associate sets of unification constraints with grammar rules
 - Constraints must be satisfied for rule to be satisfied

Feature Structures in the Grammar

For a grammar rule $\beta_0 \rightarrow \beta_1 \dots \beta_n$
 {set of constraints}

Where each constraint has one of the following forms:

- $\langle \beta_i \text{ feature path} \rangle = \text{Atomic value}$
- $\langle \beta_i \text{ feature path} \rangle = \langle \beta_j \text{ feature path} \rangle$

Example

- To enforce subject verb number agreement

S → NP VP

Only if the number of NP is equal to the number of VP



S → NP VP

<NP NUMBER> = <VP NUMBER>

Agreement in English

- We need to add PERS to our subject verb agreement constraint

This flight serves dinner.

S → NP VP

<NP AGR> = <VP AGR>

Does this flight serve dinner?

S → Aux NP VP

<Aux AGR> = <NP AGR>

- Determiner Nominal agreement can be handled similarly

These cats, This cat

NP → Det Nom

<Det AGR> = <Nom AGR>

<NP AGR> = <Nom AGR>

- And so on for other constituents and rules

How to Acquire Values

- Lexical constituents directly acquire values from the lexicon.

- Examples:

Det → this

<Det AGREEMENT NUMBER > = sg

Det → these

<Det AGREEMENT NUMBER > = pl

V → serve

<Verb AGREEMENT NUMBER > = pl

V → serves

<V AGREEMENT NUMBER > = sg

<V AGREEMENT PERSON> = 3rd

Head Features

- Observation: Features of most grammatical categories are copied from **head** child to parent (e.g. from V to VP, Nom to NP, N to Nom, ...)

- Examples:

VP → V NP

<VP AGREEMENT> = <V AGREEMENT>

NP → Det Nom

<Det AGREEMENT> = <Nom AGREEMENT>

<NP AGREEMENT> = <Nom AGREEMENT>

Nom → N

<Nom AGREEMENT> = <N AGREEMENT>

Rewrite Rules with Head Features

- Re-written as 'head' features, e.g.

VP → V NP

<VP HEAD> = <V HEAD>

NP → Det Nom

<NP HEAD> = <Nom HEAD>

<Det HEAD AGR> = <Nom HEAD AGR>

Nom → N

<Nom HEAD> = <N HEAD>

N → flights

<N HEAD AGREEMENT NUMBER> = pl

V → serves

<V HEAD AGREEMENT NUMBER> = sg

<V HEAD AGREEMENT PERSON> = 3rd

Subcategorization

- Recall: Different verbs take different types of argument
- Solution: introduce feature structures to distinguish among the various members of verb category
 - SUBCAT feature, or **subcategorization frames**

One way of doing this:

V → serves

<V HEAD AGREEMENT NUMBER> = sg

<V HEAD AGREEMENT PERSON> = 3rd

<V HEAD SUBCAT> = trans

VP → V

<VP HEAD> = <V HEAD>

<VP HEAD SUBCAT> = intrans

VP → V NP

<VP HEAD> = <V HEAD>

<VP HEAD SUBCAT> = trans

Subcategorization

- More expressive way to encode SUBCAT feature, or **subcategorization frames**

V → serves

<V HEAD AGREEMENT NUMBER> = sg

<V HEAD AGREEMENT PERSON> = 3rd

<V HEAD SUBCAT FIRST CAT> = NP

<V HEAD SUBCAT SECOND> = end

VP → V NP

<VP HEAD> = <V HEAD>

<VP HEAD SUBCAT FIRST CAT> = <NP CAT>

<VP HEAD SUBCAT SECOND> = end

Complexity in Subcategorization

- There are many phrasal types and many types of subcategorization frames
- Each verb allows many different subcategorization frames

Subcat	Example
<i>Qto</i>	asked [<i>Qto</i> "What was it like?"]
<i>NP</i>	asking [<i>NP</i> a question]
<i>Swh</i>	asked [<i>Swh</i> what trades you're interested in]
<i>Sto</i>	ask [<i>Sto</i> him to tell you]
<i>PP</i>	that means asking [<i>PP</i> at home]
<i>Vto</i>	asked [<i>Vto</i> to see a girl called Evelyn]
<i>NP Sif</i>	asked [<i>NP</i> him] [<i>Sif</i> whether he could make]
<i>NP NP</i>	asked [<i>NP</i> myself] [<i>NP</i> a question]
<i>NP Swh</i>	asked [<i>NP</i> him] [<i>Swh</i> why he took time off]

- Resources exist, e.g., COMLEX

Unification and Earley Parsing

- Unification operator: takes two feature structures and returns *either* a merged feature structure or *fail*
 - Unification algorithm goes through features in one input DAG₁ trying to find corresponding features in DAG₂ – if all match, success, else fail
- Earley Parsing with Unification:
 - Use feature structures to provide richer representation
 - Add feature structures to the grammar rules
 - Block entry into chart of ill-formed constituents
 - Add DAG to the state representation
 - New corresponding tests (e.g., completer) based on unification indicated by DAG

Summing Up

- Feature structures encodes rich information about components of grammar rules
- Unification provides a mechanism for merging structures and for comparing them
- Feature structures can be quite complex:
 - Subcategorization constraints
- Unification parsing:
 - Merge or fail
 - Modifying Earley to do unification parsing

Meaning Representation

Meaning

- So far, we have focused on the structure of language – not on what things *mean*
- We have seen that words have different meaning, depending on the context in which they are used
- Everyday language tasks that require some semantic processing
 - Answering an essay question on an exam
 - Deciding what to order at a restaurant by reading a menu
 - Realizing that you've been insulted
 - ...

2/23/2009

CSE842, Spring 2009, MSU

29

Meaning

- Now, look at **meaning representations** -- representations that link linguistic forms to knowledge of the world.
- We are going to cover:
 - What is the meaning of a word
 - How can we represent the meaning
 - What formalisms can be used

2/23/2009

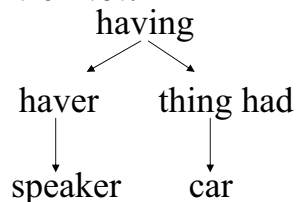
CSE842, Spring 2009, MSU

30

Common Meaning Representations

I have a car

- FOPC:
 $\exists e, y \text{ Having}(e) \wedge \text{Haver}(e, \text{Speaker}) \wedge \text{HadThing}(e, y) \wedge \text{Car}(y)$
- Semantic Net:



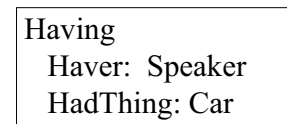
2/23/2009

CSE842, Spring 2009, MSU

31

Common Meaning Representations

- Conceptual Dependency Diagram:
Car
↑ Poss-By
Speaker
- Frame-based Representations



2/23/2009

CSE842, Spring 2009, MSU

32

Commonalities Between Representations

They all share a common foundation:

- meaning representation consists of structures composed of sets of symbols.
- all represent the meaning of a particular linguistic input
- All represent the state of affair in some world.

What Can Serve as a Meaning Representation?

- Anything that serves the core practical purposes of a program that is doing semantic processing ...
 - Answer questions (*What is the tallest building in the world?*)
 - Determining truth (*Is the blue block on the red block?*)
 - Drawing inferences (*If the blue block is on the red block and the red block is on the tallest building in the world, then the blue block is on the tallest building in the world*)
- What are basic requirements of meaning representation

What requirements must meaning representations fulfill?

- **Verifiability:** The system's ability to compare the state of affairs described by a representation to the state of affairs in some world as modeled in the knowledge base

Does Maharani serve vegetarian food?

Serves(Maharani, vegetarian food)

What requirements must meaning representations fulfill?

- **Ambiguity:** The system should allow us to represent meanings unambiguously
 - *I wanna eat someplace that's close to ICSI.*
- **Vagueness:** The system should allow us to represent vagueness
 - *I want to eat Italian food.*
(pasta? spaghetti? lasagna?)

Canonical Form

- Distinct inputs could have the same meaning
 - *Does Maharani serve vegetarian dishes?*
 - *Do they have vegetarian food at Maharani?*
 - *Are vegetarian dishes served at Maharani?*
 - *Does Maharani serve vegetarian fare?*
- Alternative:
 - Four different semantic representations
 - Problem with matching KB

2/23/2009

CSE842, Spring 2009, MSU

37

Canonical Form

- Solution: inputs that mean the same thing should have the same meaning representation
 - Vegetarian dishes, vegetarian food, vegetarian fare
 - Have, serve
- Relations among objects to be identical, how?
 - >syntactic role analysis (e.g., subjects and objects)
 - Maharani serves vegetarian dishes
 - Vegetarian dishes are served by Maharani

2/23/2009

CSE842, Spring 2009, MSU

38

Inference

- Consider a more complex request
 - Can vegetarians eat at Maharani?*
 - It would be a mistake to invoke the canonical form to force the system to assign the same representation to this request as those of:
 - Does Maharani serve vegetarian food?*
- Why are they result in the same answer?

2/23/2009

CSE842, Spring 2009, MSU

39

Inference

- Inference: system's ability to draw valid conclusions based on the meaning representation of inputs and the background knowledge
- The system must draw conclusions about the truth of propositions that are not explicitly represented in the KB, but that are logically derivable from the propositions that are present.

2/23/2009

CSE842, Spring 2009, MSU

40

Variables

I'd like to find a restaurant where I can get vegetarian food

- First observation:
 - The request does not make reference to any particular restaurant
 - Use of variables since we do not know the name of restaurant
 - A representation can be:
Serves(*X*, vegetarianFood)

Expressiveness

- Must accommodate wide variety of meanings
- FOPC is expressive enough to handle many of the NLP needs

First Order Predicate Calculus

- FOPC: provides a sound computational basis for the verifiability, inference, and expressiveness requirements
 - Supports the determination of truth
 - Supports compositionality of meaning
 - Supports representation of variables
 - Supports inference

FOPC Syntax

- **Terms:** constants, functions, variables
 - **Constants:** objects in the world, e.g. *Maharani*
 - **Functions:** concepts, e.g. *LocationOf(Maharani)*
 - **Variables:** *x*, e.g. *LocationOf(x)*
- **Predicates:** symbols that refer to relations that hold among objects in some domain or properties
 - Serves(Maharani, VegetarianFood)*
 - Restaurant(Maharani)*

FOPC Syntax

- Logical connectives permit compositionality of meaning

I only have five dollars and I don't have a lot of time

$Have(Speaker, FiveDoallars) \wedge \neg Have(Speaker, LotofTime)$

Variables and Quantifiers

Existential quantification (\exists): “There exists”,

a restaurant that serves Mexican food near ICSI

$\exists x Restaurant(x) \wedge Serves(x, MexicanFood)$

$\wedge Near((LocationOf(x), LocationOf(ICSI))$

for this logical formula to be true there must be at least one object such that if we were substitute it for the variable x, the resulting formula is true.

Variables and Quantifiers

Universal quantification (\forall): “for all”,

All vegetarian restaurants serve vegetarian food

$\forall x VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$

for this logical formula to be true the substitution of any object in the knowledge base for the universally quantifier variable should result in a true formula.