SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 98
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Artificial Intelligence and Data Science

Course Name – 19AD601 – Natural Language Processing

III Year / VI Semester

Unit 3 – SYNTACTIC ANALYSIS

Topic 9- Feature structures, Unification of feature structures.

Feature structures, Unification of feature structures.

Feature structures (FS), which contain information of different natures (morphological, syntactic and semantic) and to express the possible correlations among them.

They also allow us to refine the constraints at the lexicon level and, thus, to simplify the rules of grammar.

A feature structure is a directed graph, consisting of nodes and labelled edges.

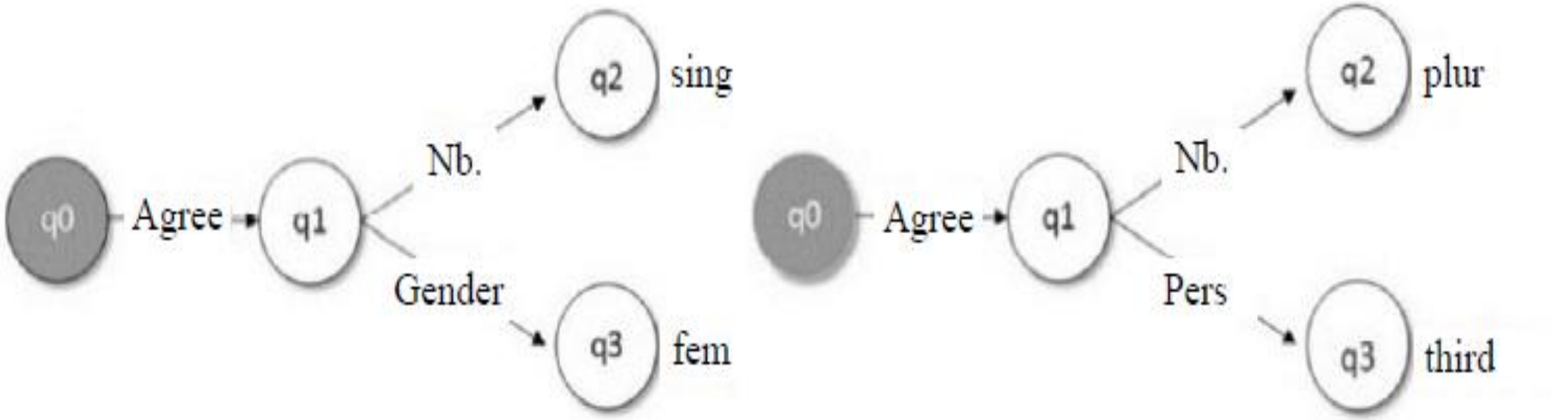
One node is special: the root node, from which every node can be reached by following edges.

A feature structure is a tuple (Q, q, δ, θ)
Q is a finite set of nodes, rooted at q
$q \in Q$ is the root node
$\theta : Q \dashrightarrow$ Type is a partial typing function
$\delta : Feat \times Q \rightarrow Q$: a partial feature value function

Feature structures, Unification of feature structures.

For example

Consider the following graph



Feature structures, Unification of feature structures.

– Q is a finite set of nodes: $Q = \{q_0, q_1, q_2, q_3\}$;

– $q' \in Q$ is the initial state of the graph $q' = q_0$. As we can note, in Figure 4.28, the initial state of this graph is colored to differentiate it from other states;

– δ represents a partial function in the graph such as: $\delta(q_0, \text{Agreement}) = q_1$, $\delta(q_1, \text{Nb.}) = q_2$, $\delta(q_1, \text{Gender}) = q_3$;

– θ represents a leaf that corresponds to a feature. Thus: $\theta(q_2)$=sing, $\theta(q_3)$=fem.

Feature structures, Unification of feature structures.

Two types of feature structures
Atomic feature structures in which all features have a simple value and

Complex feature structures (CFS) where features can have other feature structures as a value.

atomic value: an unstructured value, one with only one part

$$\begin{bmatrix} \text{tense} & \textbf{past} \\ \text{person} & \textbf{2} \end{bmatrix}$$

complex value: a structured value, itself a feature structure

$$\begin{bmatrix} \text{tense} & \text{past} \\ \text{agreement} & \begin{bmatrix} \textbf{person} & \textbf{2} \\ \textbf{number} & \textbf{singular} \end{bmatrix} \end{bmatrix}$$

Unification of feature structures.

Unification

- Unification is the operation of merging information-bearing structures, without loss of information if the unificands are consistent.

- Feature structure unification (t) is the operation of combining two feature structures so that the result is the most general feature structure that is subsumed by the two unificands (the least upper bound). If there is no such structure, then the unification fails.

- Two feature structures that can be unified are compatible (or consistent). Comparability entails compatibility, but not the other way round.

- There is untyped feature structure unification and typed feature structure unification.

- Token-identity: two feature structures are token-identical if they are the same object.
- Consistent/compatible: two feature structures are consistent if they have the same value, the values of their common features are consistent.

Feature structures, Unification of feature structures.

- In implementations, there are two ways to perform unification:
- Destructive unification: in the process of unifying two structures, one is modified and will contain the result

- Non-destructive unification: the unificands are not changed, and the result is a totally new structure.

- The former is faster, but gives undesirable effects in some cases. For instance, when you apply a grammar rule, you do not want the rule to be different after the application.

- Non-destructive unification is easier to keep track of, but requires copying. Because it does not change the feature structures, the latter is used in implementations.

THANK YOU