

Lattice and Feature Structures Assignment

Jean Mark Gawron

Linguistics

San Diego State University

gawron@mail.sdsu.edu

<http://www.rohan.sdsu.edu/~gawron>

2006 Oct 12

1 Divisibility

Consider the **infinite** set consisting of all positive multiples of some integer, say 5. Call this Mult_5 :

$$\text{Mult}_5 = \{5, 10, 15, 20, \dots, 150, 155, \dots\}$$

Consider the relation $a \mid b$ which holds when a evenly divides b :

$$a \mid b \text{ if and only if } a \text{ evenly divides } b$$

Examples:

$$\begin{aligned} 5 &\mid 10 \\ \neg 10 &\mid 5 \\ 10 &\mid 10 \\ \neg 20 &\mid 15 \\ 15 &\mid 150 \end{aligned}$$

As a warmup:

1. Consider:

$$\mathcal{A} = \langle \text{Mult}_5, \mid \rangle$$

Show \mathcal{A} is a poset. [$a \mid b$ is your relation. Show it's a weak order.]

2. Find $\text{glb}\{5, 10\}$.
3. Find $\text{lub}\{5, 25\}$.
4. Find $\text{glb}\{40, 60\}$.
5. Find $\text{lub}\{40, 60\}$.

To show that \mathcal{A} is a lattice poset, do the following (which is actually a little more than necessary):

5. Show informally that for any two $x, y \in \text{Mult}_5$, $\text{glb}\{x, y\}$ exists. [Start out by defining it!]
6. Show informally that for any two $x, y \in \text{Mult}_5$, $\text{lub}\{x, y\}$ exists.
7. Verify that Axioms L1-L4 on pp. 279-280 are satisfied.

2 Unifications

Compute the following unifications:

$$(1) \quad \begin{bmatrix} \text{to} & \boxed{1} \\ \text{through} & \boxed{2} \end{bmatrix} \sqcup \begin{bmatrix} \text{to} & \boxed{3} \\ \text{fro} & \boxed{3} \end{bmatrix}$$

(2)

$$(3) \quad \begin{bmatrix} \text{to} & \begin{bmatrix} \text{mo} & \text{a} \\ \text{larry} & \text{b} \\ \text{curly} & \text{c} \end{bmatrix} \\ \text{through} & \text{c} \\ \text{fro} & \begin{bmatrix} \text{larry} & \boxed{2} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{to} & \boxed{3} \\ \text{fro} & \boxed{3} \end{bmatrix}$$

(4)

$$(4) \quad \begin{bmatrix} \text{to} & \begin{bmatrix} \text{mo} & \text{a} \\ \text{larry} & \text{b} \\ \text{curly} & \text{c} \end{bmatrix} \\ \text{through} & \boxed{2} \\ \text{fro} & \begin{bmatrix} \text{larry} & \boxed{2} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{to} & & \boxed{3} \\ \text{through} & \text{c} & \\ \text{fro} & & \boxed{3} \end{bmatrix}$$

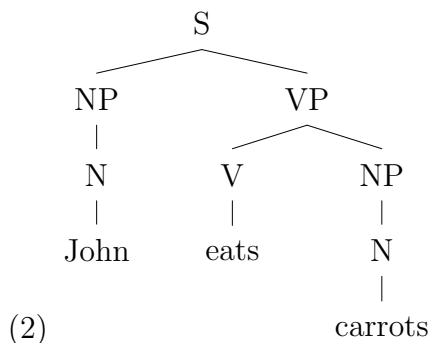
3 A feature structure grammar

Let's write a grammar. We assume a grammar assigns to each sentence a **phrase structure tree** and a **feature structure**.

For the sentence

- (1) John eats carrots.

We want the grammar to assign the phrase structure tree:



For the feature structure, let's assume we want

- (3) $\left[\begin{array}{ll} \text{cat} & \text{S} \\ \text{num} & \text{sing} \\ \text{args} & \langle \rangle \end{array} \right]$

Notice the cat is S and the args list is empty. So we define a complete sentence as one that has had all of its argument requirements satisfied. We'll try to block sentences like:

- (4) John devours.

by saying the verb *devours* imposes certain argument requirements that aren't satisfied. In effect the S we'll build will have one argument left over, so its args list won't be $\langle \rangle$, and the grammar won't **admit** it.

Let's also assume words come in a dictionary (lexicon), and feature structures assigned there. Figure 1 shows our noun lexicon. Figure 2 shows our verb lexicon. Figure 3 shows the lexicon for words that just have very simple feature structures (only syntactic category information).

carrot	$\begin{bmatrix} \text{cat} & \text{n} \\ \text{case} \\ \text{gen} \\ \text{num} & \text{sing} \end{bmatrix}$	John	$\begin{bmatrix} \text{cat} & \text{n} \\ \text{case} \\ \text{gen} \\ \text{num} & \text{sing} \end{bmatrix}$
carrots	$\begin{bmatrix} \text{cat} & \text{n} \\ \text{case} \\ \text{gen} \\ \text{num} & \text{pl} \end{bmatrix}$	sheep	$\begin{bmatrix} \text{cat} & \text{n} \\ \text{case} \\ \text{gen} \\ \text{num} \end{bmatrix}$
him	$\begin{bmatrix} \text{cat} & \text{np} \\ \text{case} & \text{acc} \\ \text{gen} & \text{masc} \\ \text{num} & \text{sing} \end{bmatrix}$	he	$\begin{bmatrix} \text{cat} & \text{np} \\ \text{case} & \text{nom} \\ \text{gen} & \text{masc} \\ \text{num} & \text{sing} \end{bmatrix}$

Figure 1: Noun Lexicon

3.1 The grammar and how it works

Our **grammar** consists of a lexicon and set of rules for building phrases out of words. These rules are given in (5):

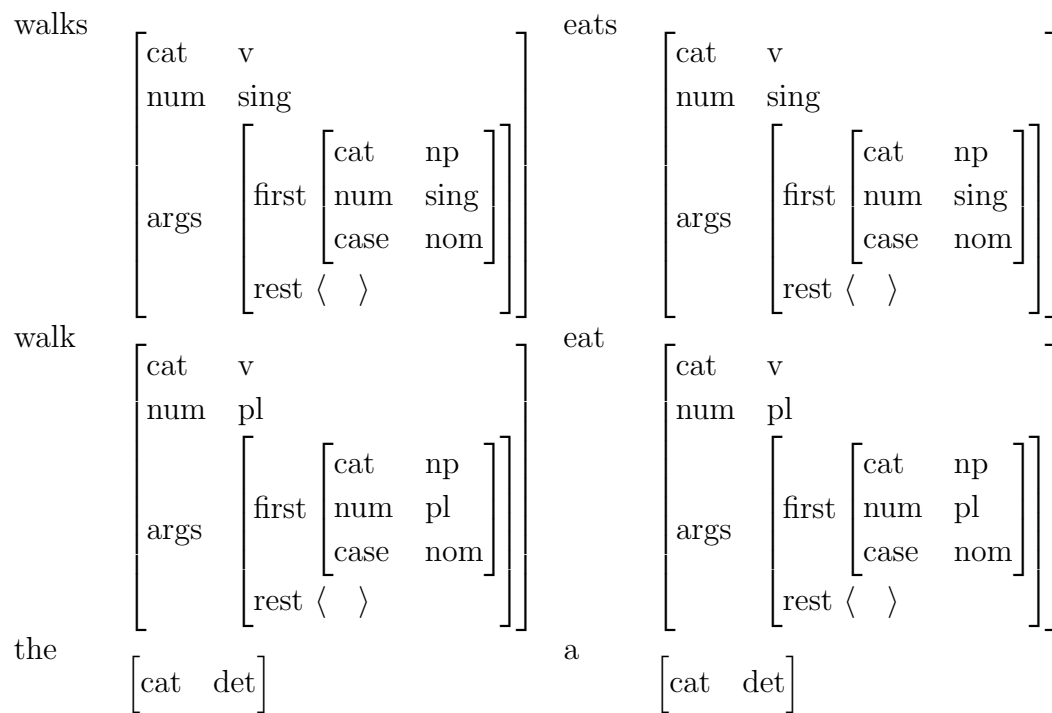


Figure 3: Rest of the Lexicon (including some verbs)

$$\begin{array}{l}
(5) \quad \hline
S \rightarrow NP VP \\
(a) \quad S (\langle \text{args} \rangle) = VP (\langle \text{args rest} \rangle) \\
(b) \quad VP (\langle \text{args first} \rangle) = NP \\
(c) \quad S (\langle \text{args} \rangle) = \langle \quad \rangle \\
(d) \quad S (\langle \text{cat} \rangle) = S \\
\hline
VP \rightarrow V NP \\
(e) \quad VP (\langle \text{args} \rangle) = V (\langle \text{args rest} \rangle) \\
(f) \quad V (\langle \text{args first} \rangle) = NP \\
\hline
VP \rightarrow V \\
(g) \quad VP (\langle \text{args} \rangle) = V (\langle \text{args} \rangle) \\
\hline
NP \rightarrow (\text{Det}) N \\
(h) \quad NP = N \\
\hline
\end{array}$$

The rules consist of a set of phrase structure rules such as

$$(6) \quad S \rightarrow NP VP$$

which tell us how to build trees like the one in (2) and some **equations** associated with each phrase structure rule which tell us how to build the **feature structure** associated with the tree. The equations associated with the S-rule in (6) are

$$\begin{array}{l}
(7) \quad (a) \quad S (\langle \text{args} \rangle) = VP (\langle \text{args rest} \rangle) \\
\quad \quad (b) \quad VP (\langle \text{args first} \rangle) = NP \\
\quad \quad (c) \quad S (\langle \text{args} \rangle) = \langle \quad \rangle \\
\quad \quad (d) \quad S (\langle \text{cat} \rangle) = S
\end{array}$$

In these equations we use **S** to stand for the feature structure for the entire sentence and we place constraints on that feature structure depending on the feature structures built for the two daughters **NP** and **VP**.

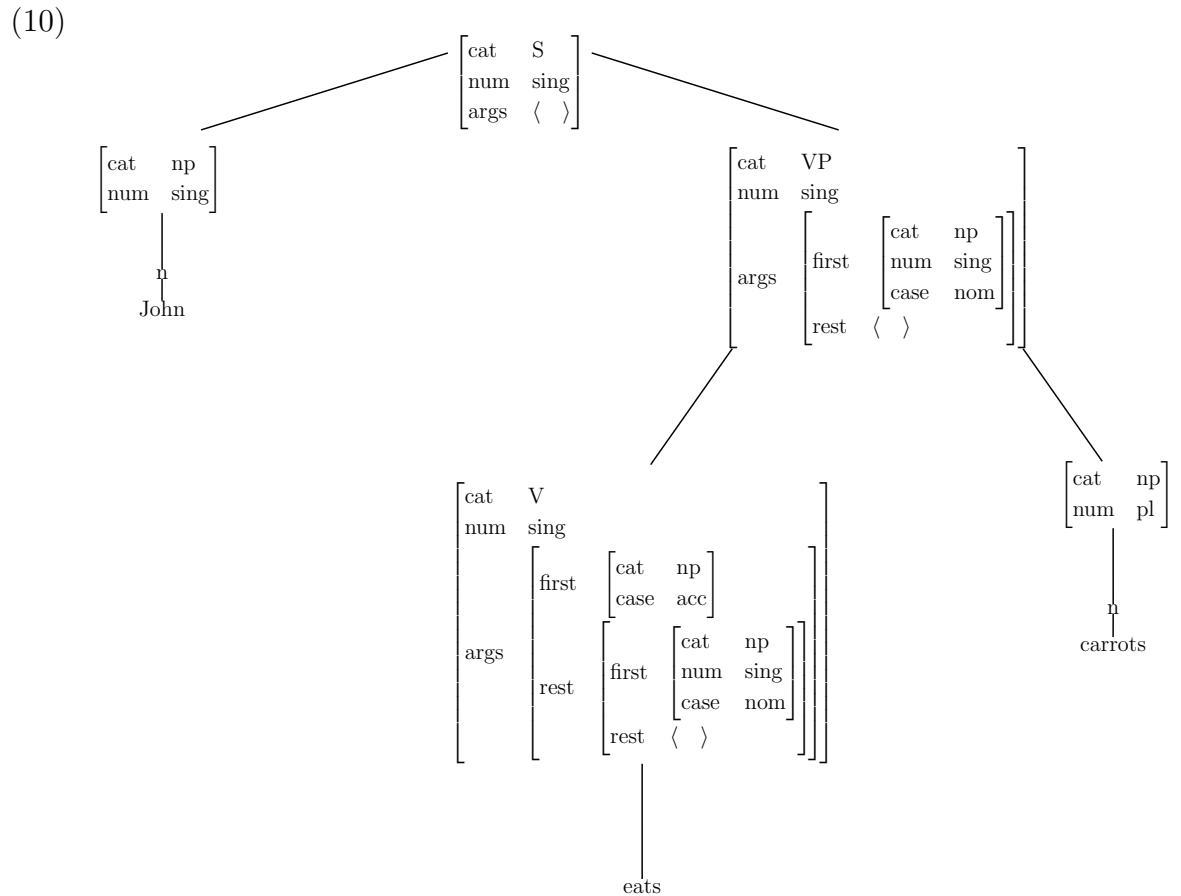
The first equation says that the S's feature structure has an **args** attribute and the value of that **args** attribute is the same as the value of the **rest** attribute of the **args** attribute of the **VP**. So if the **VP** is:

$$(8) \left[\begin{array}{cc} \text{cat} & \text{vp} \\ \text{num} & \text{pl} \\ \text{args} & \left[\begin{array}{cc} \text{first} & \left[\begin{array}{cc} \text{cat} & \text{np} \\ \text{num} & \text{pl} \\ \text{case} & \text{nom} \end{array} \right] \\ \text{rest} & \langle \ \rangle \end{array} \right] \end{array} \right]$$

Then the value of the **args** attribute for the S-feature structure is $\langle \ \rangle$. That is, the feature structure for the S contains at least:

$$(9) \left[\text{args} \ \langle \ \rangle \right]$$

These rules produce the following analysis for each node of the tree in (2):



What does all this have to do with lattices? This: When two equations place constraints on the same value in a feature structure, the results must be **unified**. For example, equation (5f), repeated here: requires that the verb's first arg unify with the f-structure of the np that follows it:

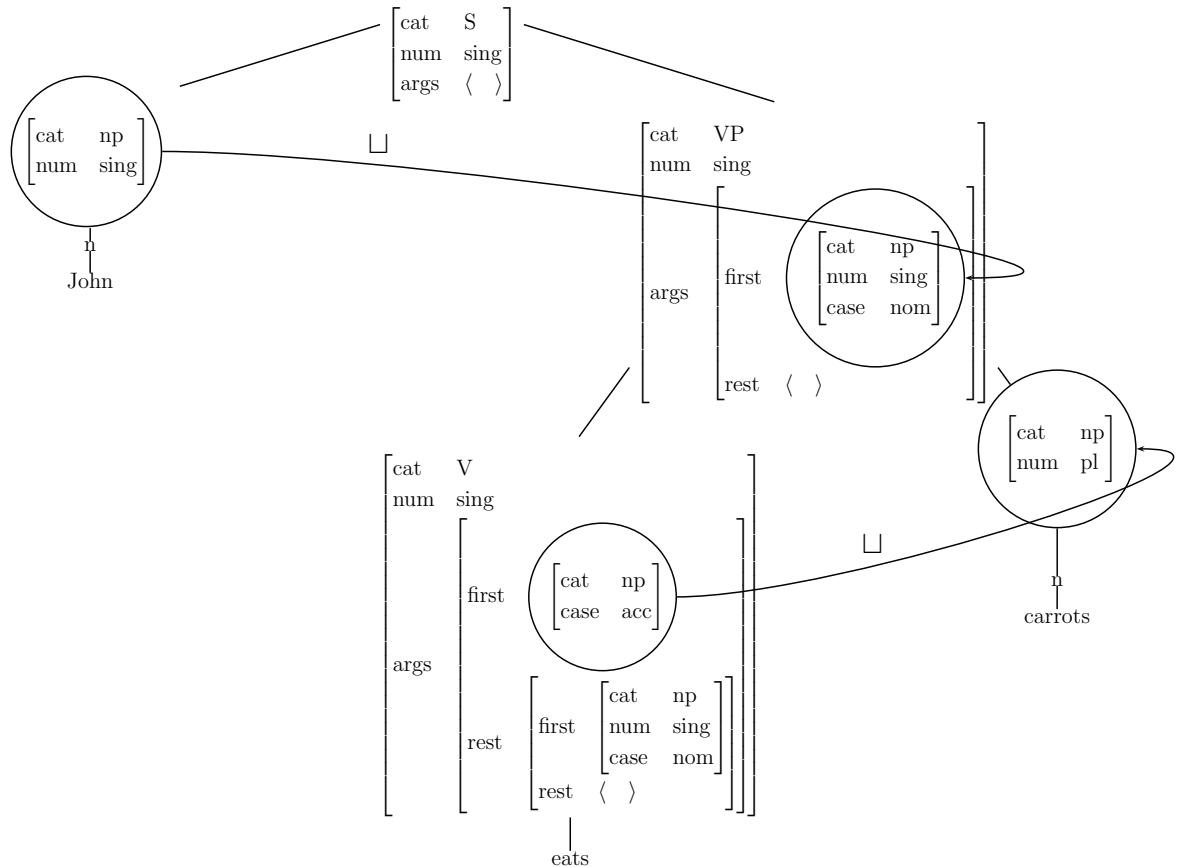
$$(11) (f) V(\langle \text{args first} \rangle) = NP$$

Equation (5b) requires that the feature-structure of the subject of the sentence unify with the vp's arg's first feature structure:

$$(12) (b) VP(\langle \text{args first} \rangle) = NP$$

The effect of these two equations is highlighted in (13):

$$(13) \text{Effect of Equations (5b) and (5f)}$$



As another example equation (5a), repeated here: requires the rest of the VP's args to unify with the S's args; equation (5e), repeated here: requires the rest of the V's args to unify with the VP's args:

mother node's arguments. By the time we reach a sentence, we should build something that has no more requirements to be satisfied. This intuition is captured by the requirement in equation (5c) that

$$S(\langle \text{args} \rangle) = \langle \ \rangle$$

That is, the args of the S is the empty list.

We will say that the grammar **fails to admit** a sentence when we can't build a feature structure for a sentence. When will that happen, since the rules tell us exactly how to build a feature structure for every tree? There will fail to be a feature structure for a tree when the result of unification is \top , because \top is not a feature structure.

Although \top is in our unification lattice (in fact it is the maximal element), it does not count as a feature structure; intuitively this is because it represents a contradictory description, rather than a satisfactory linguistic description.

When do we get \top as the result of a unification? This happens whenever we try to unify to conflicting values. For example:

$$\text{sing} \sqcup \text{pl} = \top$$

This follows directly from our definition of \sqcup .

Another thing an atomic value can't unify with is a feature structure with real values and attributes (as opposed to a variable). For example: $\langle \ \rangle$ is an atomic value.

$$\begin{aligned} \langle \ \rangle \sqcup \boxed{1} &= \langle \ \rangle \\ \langle \ \rangle \sqcup \begin{bmatrix} \text{cat} & \text{np} \\ \text{num} & \text{sing} \end{bmatrix} &= \top \end{aligned}$$

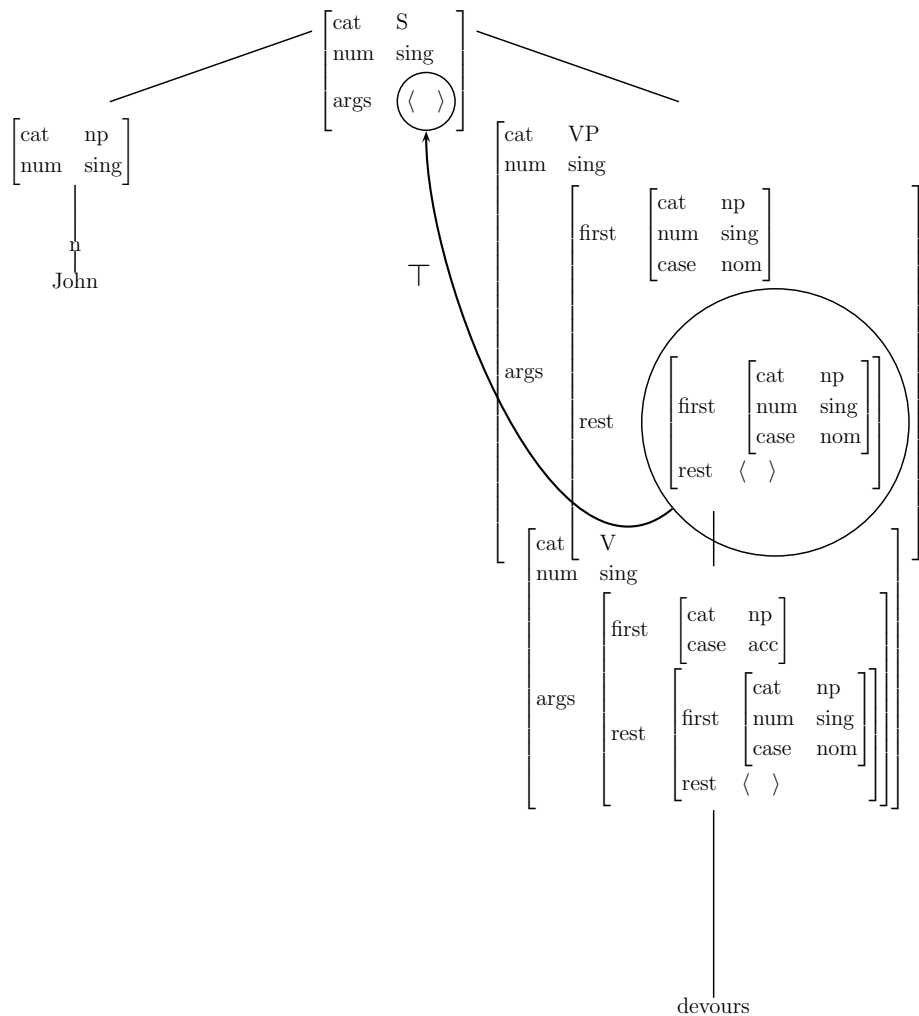
(16) gives an example of a tree without a feature. The failed unification happens right at the very top of the tree. As noted above, equation (5c) says the S-node's args list must be the empty list:

$$S(\langle \text{args} \rangle) = \langle \ \rangle$$

But the S's arg list is ALSO unified with the rest of the VP's arg list by Equation (5a):

$$S(\langle \text{args} \rangle) = \text{VP}(\langle \text{args rest} \rangle)$$

(16)



3.2 Unification grammar Exercises

1. Determine which of the following sentences are admitted by the grammar. For those admitted draw a feature structure “tree” like the one in (10). This tree should show the feature structures for both np nodes, and for the v, vp and s nodes.
 - (a) John like carrots.
 - (b) The sheep likes carrot.

- (c) The sheep like carrot.
- (d) He likes he.
- (e) Him like he.
- (f) The carrot likes him.

For sentences not admitted explain which equations in (5) are responsible for the failure.

2. Suppose we extend the grammar with following **ditransitive rule**:

$$\begin{aligned}
 \text{VP} &\rightarrow \text{V NP}_1 \text{ NP}_2 \\
 (e) \quad \text{VP} (\langle \text{args} \rangle) &= \text{V} (\langle \text{args rest rest} \rangle) \\
 (f) \quad \text{V} (\langle \text{args first} \rangle) &= \text{NP}_1 \\
 (g) \quad \text{V} (\langle \text{args rest rest first} \rangle) &= \text{NP}_2
 \end{aligned}$$

Write the correct lexical entry for a ditransitive verb like *hand*. With your entry and the rule above, the grammar should be able to admit:

(17) John handed the sheep the carrots.

Give the analysis tree for this sentence, with feature structures for the three NPs, the V, VP, and S.