

# Unification and Feature Structures

---

Remi van Trijp

Symbolic programming for non-programmarians

Erice, Italy (14 July 2007)

**Sony CSL**

Sony Computer Science Laboratory Paris

sony computer science laboratory Paris

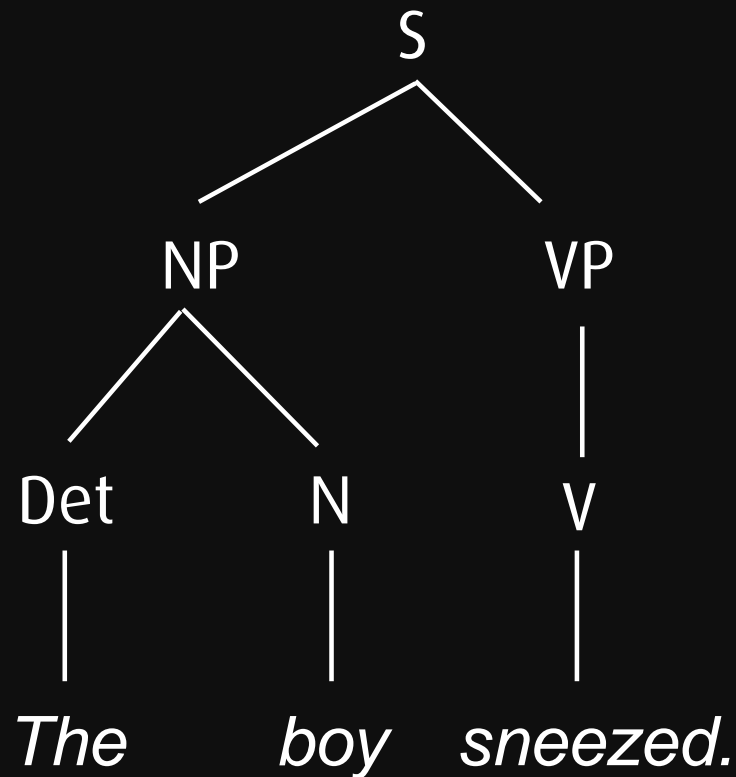
## Good sources

- Unification (and A.I. Programming)
  - Norvig, Peter (1992). **Paradigms of Artificial Intelligence Programming.** Case Studies in Common Lisp. San Francisco: Morgan Kaufmann.
- Feature structures (& Computational Linguistics)
  - Jurafsky, Daniel & James H. Martin (2000). **Speech and Language Processing.** An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. New Jersey: Prentice Hall.

# Contents

- Why feature structures and unification?
- Why construction grammar?
- How can we apply feature structures and unification in (computational) construction grammar?

# A simple grammar of English



$S \rightarrow NP VP$

$NP \rightarrow Det N$

$VP \rightarrow V$

$Det \rightarrow a, the, this, these$

$N \rightarrow boy, cat, dog, women$

$V \rightarrow sneezed, shouted$

'(S (NP (DET "the") (N "boy"))) (VP (V "sneezed")))

# Demo of the Simple Grammar

- See file "erice-1.lisp"

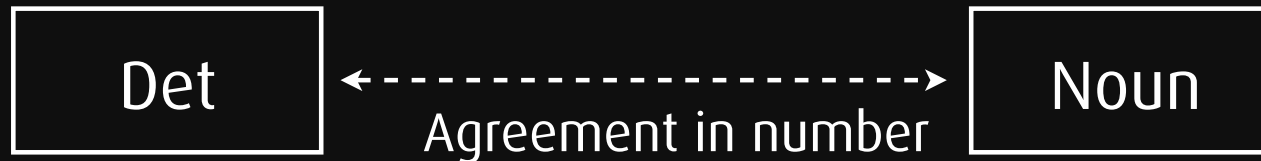
## More refined model...

- "These cat shouted."
- Creating lots of new categories?
  - E.g. Singular-Noun vs Plural-Noun, etc.
  - Also lots of new rules needed
    - > NP -> Singular-Det Singular-Noun  
/ Plural-Det Plural-Noun
- Alternative: constraint-based formalism
  - NP -> Det N  
only if the grammatical number of Det  
is equal to the grammatical number of N

# More refined model...

## NP-construction

Form: Det > Noun



# Feature Structures

- Treat categories as objects with complex sets of properties associated with them

Feature-1	Value-1
Feature-2	Value-2
...	
Feature-n	Value-n

woman

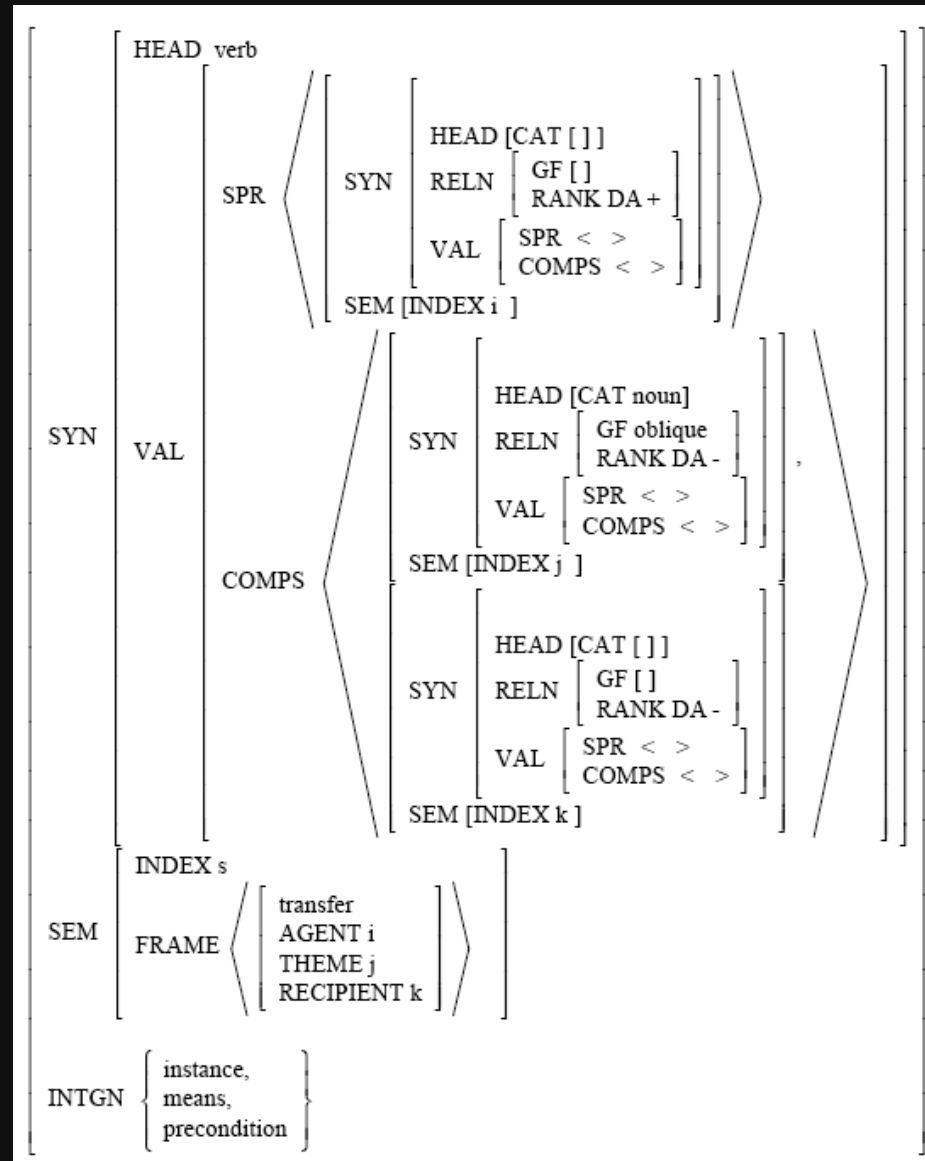
Syn-cat	noun
Number	SG
Person	3

women

Syn-cat	noun
Number	PL
Person	3

# Feature Structures

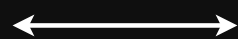
(from L. Michaelis)



# Implementing Feature Structures

woman

Syn-cat	noun
Number	SG
Person	3



'(woman

(syn-cat noun)  
(number SG)  
(person 3))

women

Syn-cat	noun
Number	PL
Person	3



'(women

(syn-cat noun)  
(number PL)  
(person 3))

# Why unification?

- Unification is a powerful kind of pattern-matcher...

- (unifier '(number PL) '(number PL))

==> '(number PL)

- (unifier '(number PL) '(number SG))

==> FAILS!!

# Why unification?

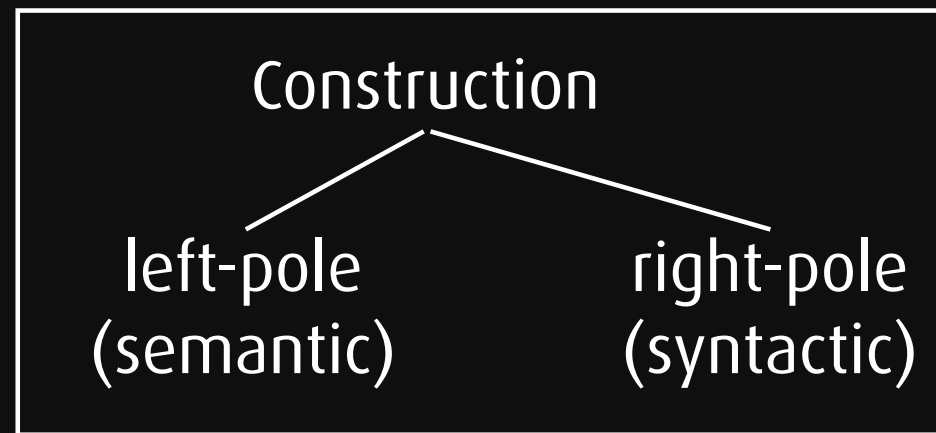
- Also two patterns which contain logic variables can be matched against each other...
- Variables are marked by ? (question mark)
- ```
> (unify '(+ ?x 1) '(+ 2 ?y))
```

```
=> ((?Y . 1) (?X . 2))
```
- Demonstration (erice-2.lisp)

# Only syntactic feature structures?

- “Colourless green ideas sleep furiously.”  
(Noam Chomsky)
- syntactically correct, but makes no sense...
- Construction Grammar: form-meaning mappings
- **coupled** feature structures: mappings from form to meaning and vice versa



# Fluid Construction Grammar

- Incorporated in Babel2 (general cognitive framework)

<http://emergent-languages.org>

- Also see the reader for the most important papers
- A (brief) preview of the class of Joachim De Beule...

# A production example

- Suppose a speaker wants to produce an utterance for the following meaning:<sup>(1)</sup>

((see event-1 true)  
(see-1 event-1 Jack)  
(see-2 event-2 Jill)  
(unique-person Jack)  
(unique-person Jill))

[ Jack sees Jill. ]

(1) See the poster presentation by Wouter Van den Broeck for more on conceptualization and semantics.

# A production example

## Coupled feature structure

Semantic pole:

```
((sentence  
  (meaning  
    ((see ev-1 true)  
     (see-1 ev-1 Jack)  
     (see-2 ev-1 Jill)  
     (unique-person Jack)  
     (unique-person Jill))))))
```

<-->

Syntactic pole:

```
((sentence))
```

Semantic pole:

```
((?unit-x  
  (meaning  
    (== (see ?ev true)  
        (see-1 ?ev ?obj-1)  
        (see-2 ?ev ?obj-2))))))
```

Rule-see

<-->

Syntactic pole:

```
((?unit-x  
  (syn-subunits (== ?unit-y)))  
 (?unit-y  
  (form ((stem ?unit-y "see")))  
  (syn-cat v) (agr ?agr)))
```

merge

# A production example

## Coupled feature structure

Semantic pole:

<-->

Syntactic pole:

```
((sentence
  (sem-subunits (verb-unit))
  (meaning
    ((unique-person Jack)
     (unique-person Jill))))
(verb-unit
  (meaning
    ((see ev-1 true)
     (see-1 ev-1 Jack)
     (see-2 ev-1 Jill))))))
```

```
((sentence
  (syn-subunits (verb-unit)))
(verb-unit
  (syn-cat v)
  (form
    (== (stem verb-unit "see"))))
(agr ?agr)))
```

Demo: erice-3.lisp

## Summary

- Feature structures provide flexible and fine-grained ways to represent (linguistic) information;
- Unification is a powerful constraint-based approach to processing these feature structures;
- Fluid Construction Grammar implements form-meaning mappings through coupled feature structures;
- Unification is used for matching and merging these structures.

# Thank you for your attention

- Wouter Van den Broeck  
[wouter@csl.sony.fr](mailto:wouter@csl.sony.fr)

Remi van Trijp  
[remi@csl.sony.fr](mailto:remi@csl.sony.fr)

- <http://emergent-languages.org>  
<http://www.csl.sony.fr/>

- Slides and files will be made available at:

<http://www.csl.sony.fr/erice2007/>