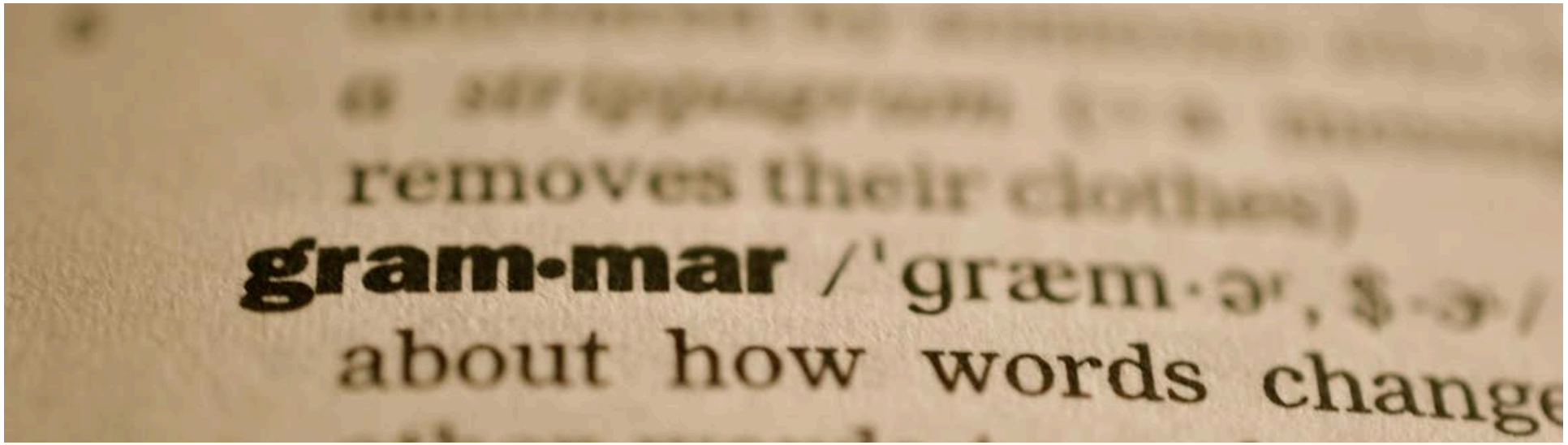


Fluid Construction Grammar

Tutorial Lecture 2.2:
Argument Structure and Coercion

Remi van Trijp
Sony Computer Science Laboratory Paris
remi@csl.sony.fr

Today's Lecture

- Argument structure:
  - Expressing the relations between events and the participants that play a role in those events
  - Interaction between verbs and argument structure constructions
- Exercises/Hands-on examples: lecture-2_02.lisp

Also see:

- "Goldberg, Adele (1995). *Constructions: A Construction Grammar Approach to Argument Structure.* The University of Chicago Press." (and subsequent work)
- Martin Hilpert's introduction at https://www.youtube.com/watch?v=GN1MrP396Y4

Part 1

VERBS AND ARGUMENT STRUCTURE CONSTRUCTIONS

Verbal Meanings

- Verbs are events that involve participants
  - Push < pusher pushed >

Verbal Meanings

- Verbs are events that involve participants
  - Push < pusher pushed >

Verbal Meanings

- Verbs are events that involve participants
  - Push < pusher pushed >

((push ?ev) (pusher ?ev ?x) (pushed ?ev ?y))

Building a Verbal Construction

- Verbs are events that involve participants
  - push < pusher pushed >

((push ?ev) (pusher ?ev ?x) (pushed ?ev ?y))

-
```
(def-lex-skeleton pushed-lex
      :meaning  (== (push ?ev)
                    (pusher ?ev ?x)
                    (pushed ?ev ?y))
      :args (?ev)
      :string "pushed")
```

Building a Verbal Construction

- (def-lex-cxn pushed-lex

```
(def-lex-skeleton pushed-lex
        :meaning  (== (push ?ev)
                      (pusher ?ev ?x)
                      (pushed ?ev ?y))
        :args (?ev)
        :string "pushed")

(def-lex-cat pushed-lex
        :sem-cat  ((sem-class event))
        :syn-cat  ((lex-class verb))))
```

Lexical and Phrasal Constructions

- In the supporting Lisp-file, you will find a number of constructions that are equivalent to the following phrase structure rules:

NP	→	Art Noun
VP	→	Verb
Art	→	“the”
Noun	→	“ball”, “puppet”
Verb	→	“pushed”

- If you do not understand these constructions, please revise the lectures on lexical and phrasal constructions

Lexical and Phrasal Constructions

- In the supporting Lisp-file, you will find a number of constructions that are equivalent to the following phrase structure rules:

S	→	NP VP NP
NP	→	Art Noun
VP	→	Verb
Art	→	“the”
Noun	→	“ball”, “puppet”
Verb	→	“pushed”

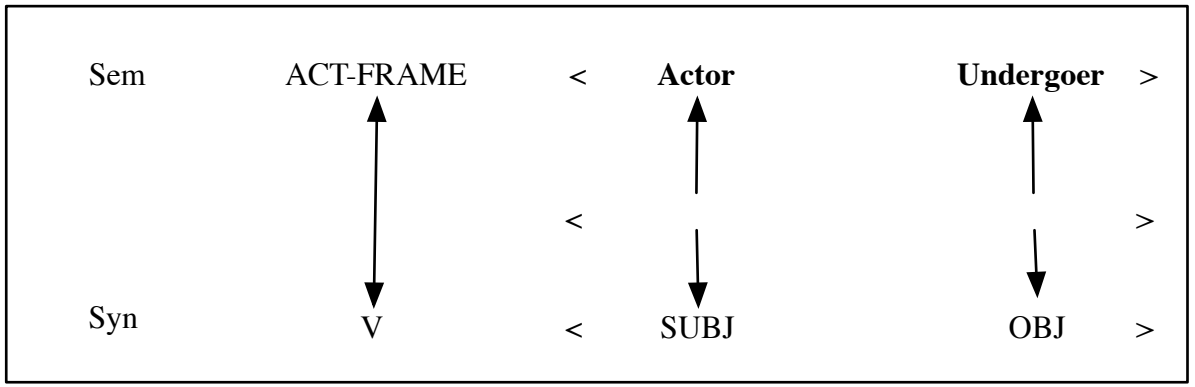
- If you do not understand these constructions, please revise the lectures on lexical and phrasal constructions

Lexical and Phrasal Constructions

- Test what happens if you do not have argument structure constructions
- The FCG-system still comes up with a solution, but…
  - The final structure is not the desired linguistic analysis of the sentence
  - In parsing, meanings are **disconnected** (i.e. we do not know who did what to whom)
  - In production, there is no basis for choosing acceptable word order and word forms

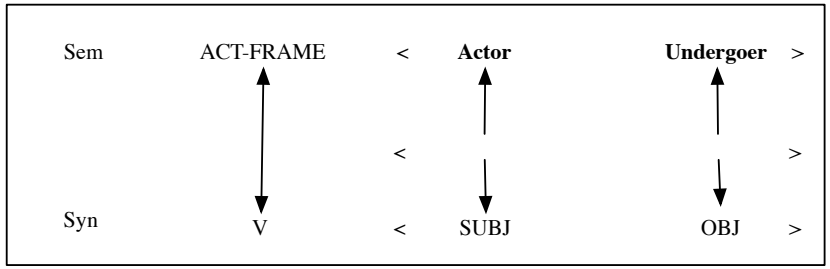
Argument Structure Constructions

- For the time being, you can think of argument structure constructions as rewrite rules for the symbol S:
  e.g. S → NP VP NP
- But really they are **meaningful** mappings between **argument roles** and **grammatical functions**



The Syntactic Pole of the Active-Transitive Construction

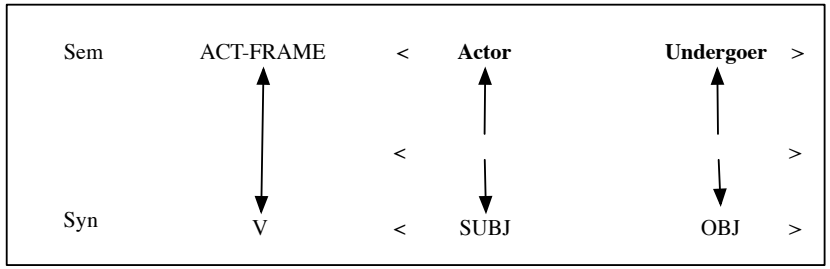
S → NP VP NP



- ((?VP
  (footprints (==0 arg-cxn))
  (syn-cat (==1 (phrase-type VP)
  (head ?vp-head)))
  (syn-valence (==1 (subject ?Subject)
  (object ?object))))
  (?Subject
  (syn-cat (==1 (phrase-type NP)
  (head ?subject-head)
  (syn-role subject))))
  (?Object
  (syn-cat (==1 (phrase-type NP)
  (head ?object-head)
  (syn-role object))))
  (root
  (form (== (precedes ?subject-head ?VP-head)
  (precedes ?VP-head ?object-head))))
  ((J ?VP) (footprints (==1 arg-cxn)))
  ((J ?transitive-clause)
  (syn-subunits (?subject ?vp ?object))
  (syn-cat ((clause-type transitive)))))

The Semantic Pole of the Active-Transitive Construction

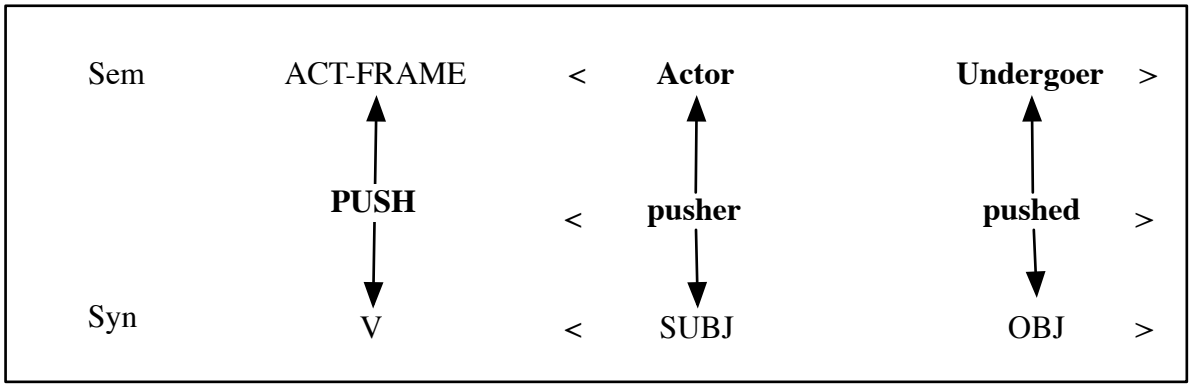
S → NP VP NP



- ((?VP
    (footprints (==0 act-cxn))
    (sem-cat (==1 (sem-function predicating-expression)))
    (sem-valence (==1 (act-frame ?ev)
                      (actor ?ev ?x)
                      (undergoer ?ev ?y))))
  (?Subject
    (sem-cat (==1 (sem-function referring-expression)
                  (sem-role actor)))
    (args (?x)))
  (?Object
    (sem-cat (==1 (sem-function referring-expression)
                  (sem-role undergoer)))
    (args (?y))))
  ((J ?VP) (footprints (==1 arg-cxn)))
  ((J ?transitive-clause)
    (sem-subunits (?subject ?vp ?object))
    (sem-cat ((sem-function proposition)))))

Interaction between Constructions

- Somehow, the event-specific participant structure of a verb must interact with the more schematic argument structure constructions



- "Fusion": Goldberg (1995:50) writes that participant roles can be "fused" with argument roles if they are conventionally associated with a particular argument frame, and if they are semantically compatible

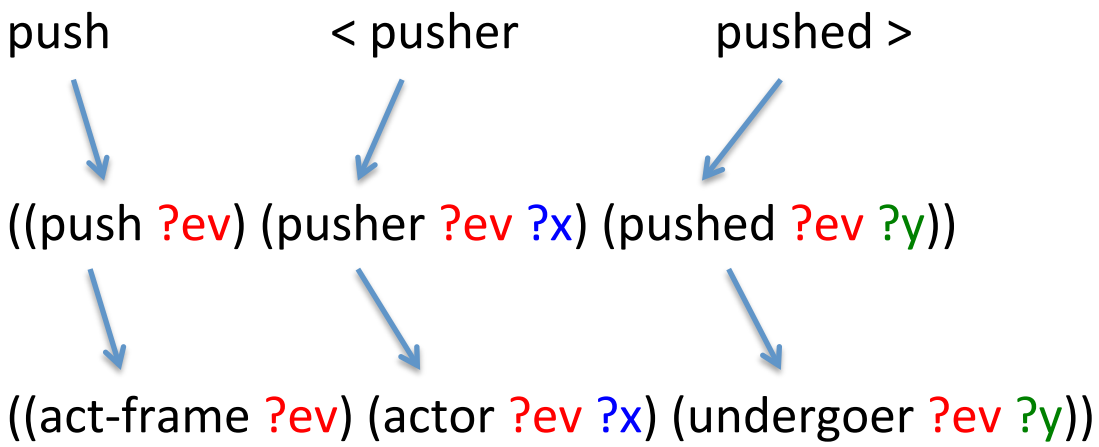
Fusion (1)

- A dynamic implementation of fusion can be operationalized in the following steps:
  - Expand the verb's event structure
  - Expand the argument structure construction's argument structure
  - Test for semantic compatibility

- E.g. analogical reasoning over events
  - van Trijp, Remi (2014). *The Evolution of Case Grammar*. Berlin: Language Science Press. (Chapter 3)
  - In this book, dynamic fusion is used for learning and inventing new semantic roles

Fusion (2)

- A conventional implementation of fusion uses the notion of **potential valence**
  - Keep a list of compatible argument roles in the verb
  - Use variable equalities for specifying which participant roles may be fused with which argument roles



Pushed-cxn Revisited

-
```
(def-lex-cxn pushed-lex

 (def-lex-skeleton pushed-lex
           :meaning (== (push ?ev)
                        (pusher ?ev ?x)
                        (pushed ?ev ?y))
           :args (?ev)
           :string "pushed")

 (def-lex-cat pushed-lex
           :sem-cat ((sem-class event))
           :syn-cat ((lex-class verb))))

 (def-lex-valence pushed-lex
           :sem-valence   ((act-frame ?ev)
                           (actor ?ev ?x)
                           (undergoer ?ev ?y))
           :syn-valence   ((subject ?subject)
                           (object ?object)
                           (oblique ?oblique))))
```

Part 2

MULTIPLE ARGUMENT REALIZATION

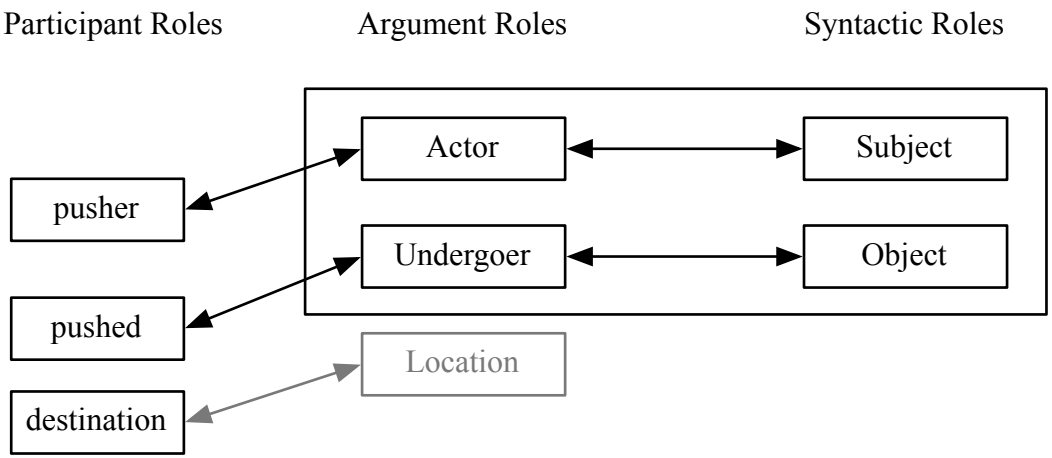
Multiple Argument Realization

- The puppet pushed the block.
- The puppet pushed the block off the table.
- The block was pushed (off the table) (by the puppet).
- …

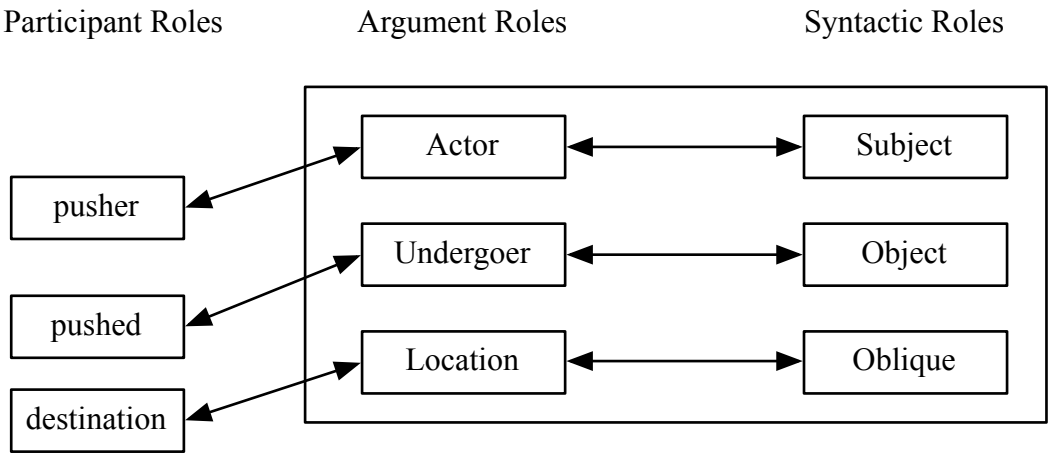
Pushed-cxn Revisited

-
```
(def-lex-cxn pushed-lex

 (def-lex-skeleton pushed-lex
            :meaning   (== (push ?ev)
                           (pusher ?ev ?x)
                           (pushed ?ev ?y))
                           (push-source ?ev ?z))
            :args (?ev)
            :string "pushed")

 (def-lex-cat pushed-lex
            :sem-cat   ((sem-class event))
            :syn-cat   ((lex-class verb))))

 (def-lex-valence pushed-lex
            :sem-valence  ((act-frame ?ev)
                           (actor ?ev ?x)
                           (undergoer ?ev ?y)
                           (location ?ev ?z))
            :syn-valence  ((subject ?subject)
                               (object ?object)
                               (oblique ?oblique))))
```

Argument Structure Constructions Select what They Need



Active-Caused Motion Construction
The puppet pushed the block.



Active-Caused Motion Construction
The puppet pushed the block off the table.

Argument Structure Constructions Select what They Need

- Technically, this "selection" is achieved through **matching**
-
```
(unify '(==1 (actor ?ev ?x)
             (undergoer ?ev ?y))
       '((actor ev-1 the-puppet)
         (undergoer ev-1 the-block)
         (location ev-1 off-the-table)))

==> ((?x . the-puppet)(?y . the-block)
     (?ev . ev-1))
```

Part 3

COERCION

Coercion

- Coercion occurs when there is a “mismatch” between the verb and the argument structure construction
  - She sneezed the napkin off the table.

Coercion

- “sneeze” is unlikely to have a destination-role in its semantic valence…
- (def-lex-cxn sneezed-lex

```
(def-lex-skeleton sneezed-lex
          :meaning (==   (sneeze ?ev)
                         (sneezer ?ev ?x))
          :args (?ev)
          :string “sneezed”)

(def-lex-cat sneezed-lex
          :sem-cat ((sem-class event))
          :syn-cat ((lex-class verb))))

(def-lex-valence sneezed-lex
          :sem-valence   ((actor ?ev ?x))
          :syn-valence   ((subject ?subject))))
```

Coercion

- Technically, we can exploit the distinction between **matching** and **merging** for allowing coercion
- E.g. the Caused-Motion Construction could only require the actor role for matching:
  (sem-valence
  `(==1 (actor ?ev ?x)))`
- And stipulate the other roles in a J-unit (which is always merged):
  (sem-valence
  `(==1 (undergoer ?ev ?y)`
  `(location ?ev ?z)))`

Summary

- Verbs and argument structure constructions
  - Verbs store potential semantic and syntactic valence
  - Argument structure constructions select the valence they need…
  - … or may coerce the verb in its pattern if the matching requirements are satisfied

Further reading:

- van Trijp, Remi (2011). A design pattern for argument structure constructions. In: Steels, Luc (ed.), *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins. www.fcg-net.org