n-PLACE RELATIONS IN NATURAL LANGUAGE

Fred Landman
Tel Aviv University
2008

1. INTRODUCTION

One place predicates: Productive
-intransitive verbs
Fred *swims* **SWIM**[1](Fred)
1
-copular predicates, like adjecival predicates
Dafna *is clever* **CLEVER**[1](Dafna)
1

If we assume a Davidsonian analysis, with an implicit event argument, the whole story goes one place up. I will still count this as a **one-place predicate** (for the moment). Thus, in our count, we ignore implicit parameters.

Fred *swims* **SWIM**[1](e,Fred)
1 (e is a swimming of Fred)

Two place predicates: Productive
-transitive verbs
Fred *kisses* Susan **KISS**[2](Fred, Susan)
1 2

-relational adjectival predicates
Fred *is proud of* Dafna **PROUD-OF**[2](Fred, Dafna)
1 2
-comparatives
Susan *is taller than* Dafna **TALLER THAN**[2](Susan, Dafna)
1 2

Three place predicates: A small (but not very small) number
-ditransitive verbs:

Fred *introduces* John to Mary **INTRODUCE**[3](Fred, John, Mary)
1 2 3

Fred *gives* Susan *Ulysses* **GIVE**[3](Fred, Susan, *Ulysses*)
1 3 2

Four place predicates: Vanishingly small, if present at all

-tri-transitive verbs
Possibly: *trade* in: John *trades* Mary the book for a cd **TRADE**^{4}(J, M, B, C)
1 2 3 4

There are, apparently, languages in which there are real tri-transitive verbs (with the arguments expressed without prepositions.)

In most languages, the lexicon doesn't like to count beyond three:
-three-place predicates are allowed.
-one-place predicates and two-place predicates are dominent.
-many productive operations for forming new one place and two place relations
(two-place → one-place: passive, reflexive, etc…
one-place → two place: comparatives)

Logic (since Frege): the theory generalizes unproblematically to n-place predicates:

> If $t_1,\ldots,t_n$ are terms and P an n-place relational constant,
> then $P(t_1,\ldots,t_n)$ is a formula.

Why not in natural language?

I will not even try to answer this question, but I will argue that the matter is **not coincidental or arbitrary**.

In this talk:

Serial verb construction in Dutch:
-Mechanism for forming n-place relations.
-Used productively to make new 3 or 4 place relations
-Beyond 4-place the serial verbs: processing difficulties.

-Drawing conclusions about other constructions for which linguistis, logicians and/or computer scientists have proposed semantic analyses involving n-place relations.

First part of this talk: 'tutorial' on the syntax of Dutch serial verbs.

2. THE SYNTAX OF SERIAL VERBS IN DUTCH

Relevant class of verbs: *let, help, see, hear,… make* (the latter only in English).

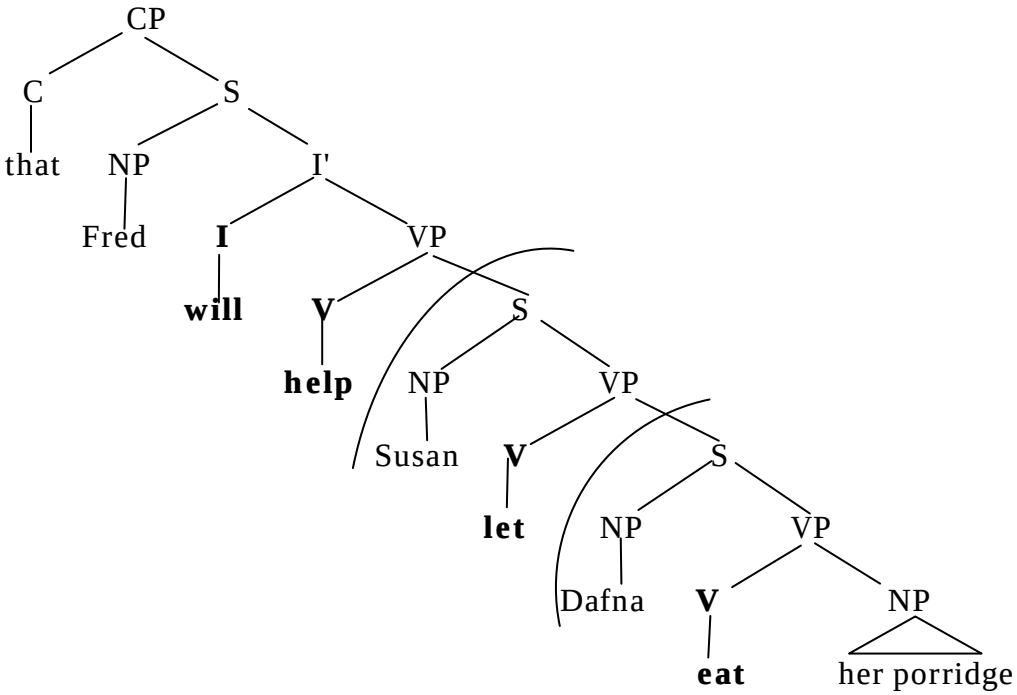
A standard assumption is that these verbs take, what are called, **small clause complements**, clauses without inflection marking (like tense or infintive marker *to*):

(1) Susan will **let** *Dafna eat her porridge*.

The construction is recursive:

(2) Fred will **help** Susan **let** Dafna eat her porridge

English structure:

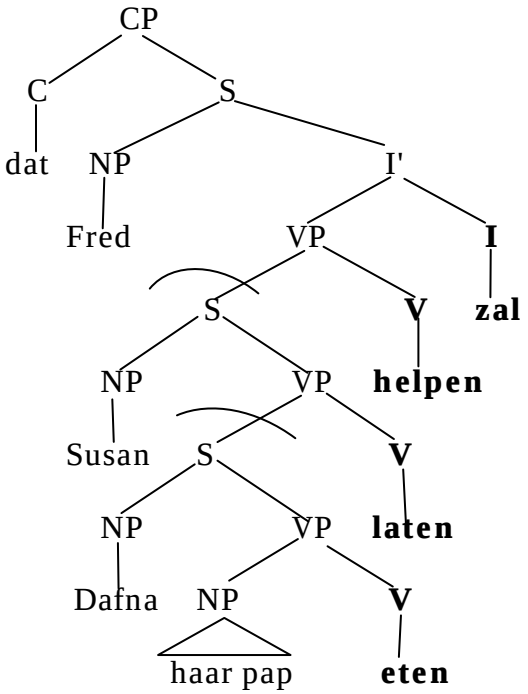


…that Fred will [**help Susan** [***let Dafna*** [**eat her porridge**]]].

The trees represent **constituent structure**:
we call any sub-tree with non-empty yield a **constituent** of the tree.

CP	complementizer phrase	NP:	noun phrase
C:	complementizer		
S:	sentence (= IP: inflection phrase)	VP:	verb phrase
I':	inflection projection	V:	verb
I:	inflection		

Dutch (and German): verbal heads (the elements I and V) are **to the right of their complements** (not left, as in English).
Flip the heads to the right in the English structure (and fill in Dutch cognates):



(3)...*dat Fred [Susan [Dafna [haar pap **eten**] **laten**] **helpen**] **zal**
that Fred Susan Dafna her porridge eat let help will

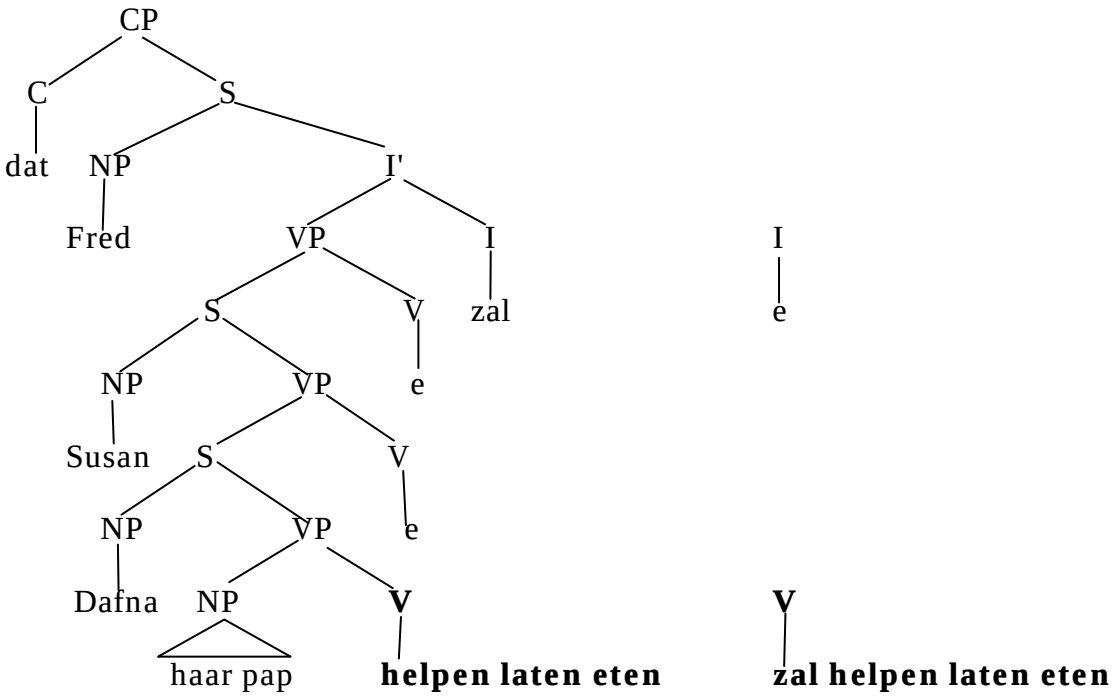
This is **ungrammatical** in Dutch:
The order of the infinitives is the **inverse order**:

(4) ...dat Fred Susan Dafna haar pap **helpen laten eten zal**.

The syntactic fact that this tutorial aims to convince you of is:

THE SERIAL VERB ASSUMPTION:

The infinitive sequence **helpen laten eten** behaves like a single verb, a **serial verb**, that sits at the lowest verb position in the tree:



... dat Fred [Susan [Dafna [haar pap **helpen laten eten**] e] e] **zal**
that Fred Susan Dafna her porridge **help let eat** will

3. THE SYNTACTIC EVIDENCE

Fact 1. In subordinated clauses with a tensed auxiliary and an infinitive, the auxiliary occcurs directly before the infinitive or directly after.

(5) a. Dat Dafna haar pap **zal** *eten*.
b. Dat Dafna haar pap *eten* **zal**
That Dafna will eat her porridge

(6) a. Dat Fred Susan Dafna haar pap **zal** [*helpen laten eten*]
b. Dat Fred Susan Dafna haar pap [*helpen laten eten*] **zal**

That Fred will help Susan let Dafna eat her porridge

Fact 2. Verbal arguments and adjuncts cannot can come between the auxiliary and the infinitive:

The infinitve sequence behaves in this way just like a single verb behaves:

(7) a. Dat Dafna haar pap **zal** **eten** her porridge **will eat**
*Dat Dafna **zal** haar pap **eten** ***will** her porridge **eat**
b. Dat Dafna haar pap rustig **zal** **eten** quietly **will eat**
*Dat Dafna haar pap **zal** rustig **eten** ***will** quietly **eat**
quietly

(8) a. Dat Fred Susan Dafna haar pap **zal** **[helpen laten eten]**
*Dat Fred Susan Dafna **zal** haar pap **[helpen laten eten]**

b. Dat Fred Susan Dafna haar pap rustig **zal** **[helpen laten eten]**
*Dat Fred Susan Dafna haar pap rustig **zal** rustig **[helpen laten eten]**

Fact 3. Verbal arguments and adjuncts cannot occur inside the verb; the inflected verb cannot occur inside the sequence of infinitives (without *te*).

The infinitive sequence behaves like a unit:

(9) a. Dat Fred Susan Dafna haar pap **zal [helpen** **laten** **eten]**
*Dat Fred Susan Dafna **zal [helpen** haar pap **laten** **eten]**
*Dat Fred Susan Dafna **zal [helpen** **laten** haar pap **eten]**

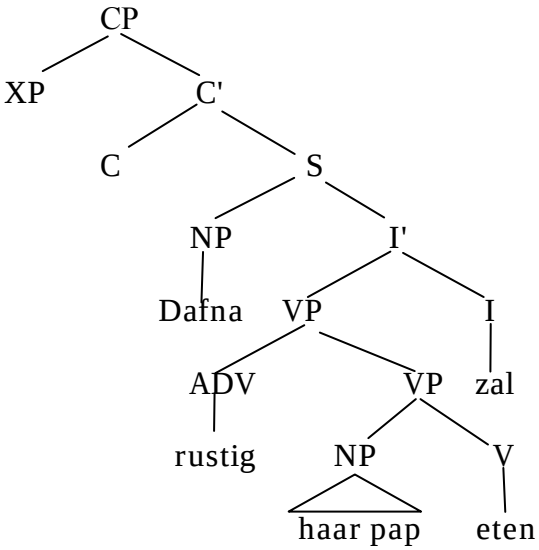
b. Dat Fred Susan Dafna haar pap rustig **zal [helpen** **laten** **eten]**
*Dat Fred Susan Dafna haar pap **zal [helpen** rustig **laten** **eten]**
*Dat Fred Susan Dafna haar pap **zal [helpen** **laten** rustig **eten]**

c. Dat Fred Susan Dafna haar pap *zal* **[helpen** **laten** **eten]**
*Dat Fred Susan Dafna haar pap **[helpen** *zal* **laten** **eten]**
*Dat Fred Susan Dafna haar pap **[helpen** **laten** *zal* **eten]**

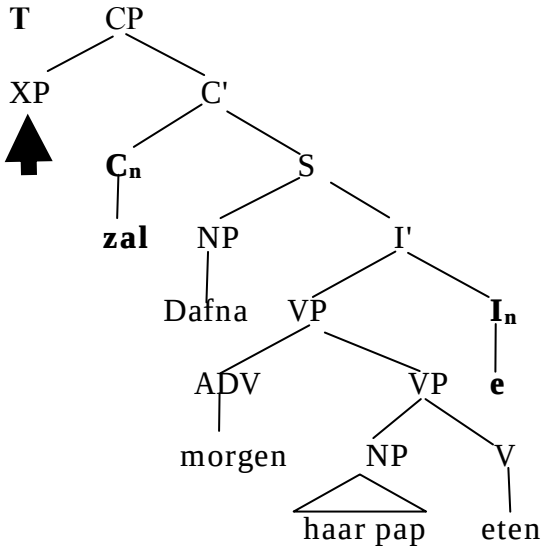
(There are some differences between different kinds of infinitives, and differences between variations here between Dutch and Flemish dialects here. The issue is interesting, but tangential to my problems.)

These facts are telling; the next set of facts is convincing. These concern verb second.

VERB-SECOND AS A TEST FOR CONSTITUENCY:



In main-clauses, the tensed verb occurs in the complementizer position C and, if the structure is indicative (not a question), the XP position must be lexically filled:



... **zal** Dafna morgen haar pap eten
will Dafna tomorrow her porridge eat

Tomorrow Dafna will eat her porridge.

T is not form a well-formed main-clause indicative, since XP, is not lexically filled.

Yield(**T**): e **zal** Dafna rustig haar pap eten.
↑

To form a felicitous indicative, some sub-string of the yield must occur in the position of e, **instead**.of where it occurs. What are the constraints?

Interestingly enough, **hardly any**:

The verb-second effect:

1. α must be the yield of a **constituent T_{α}**, a sub-tree of T.
2. **Moving T_{α}** from its position in **T** to the position XP **should not violate syntactic constraints**.

1. Lexical constituents in first position are felicitous:

(10) a. **Dafna** *zal* morgen haar pap eten.
b. **Morgen** *zal* Dafna haar pap eten.
c. **Haar pap** *zal* Dafna morgen eten.
d. **Eten** *zal* Dafna morgen haar pap .

2. Complex constituents in first position are felicitous:

(11) a. **Haar pap eten** *zal* Dafna morgen.
b. **Morgen haar pap eten** *zal* Dafna.

3. Non-constituents in first position are not felicitous:

(12) a. ***Morgen haar pap** *zal* Dafna e e eten.
b. ***Morgen – eten** *zal* Dafna e haar pap e.

4. Moving constituents to first position while violating syntactic constraints is not felicitous:

(13) a. **Haar pap** *zal* Dafna morgen eten.
b. ***Haar** *zal* Dafna morgen **pap** eten.
c. ***Pap** *zal* Dafna morgen **haar** eten.

(Moving a determiner or a noun out of a (definite) noun phrase violates syntax)

(14) ***Dafna morgen pap eten** *zal*.

(The constituent with topnode S: **Dafna morgen pap eten** contains the syntactic variable e_n of the tensed verb in C_n. It violates syntax to put that variable in a higher position than syntactic operatorC_n.)

We look at the verb cluster and put the tensed auxiliary verb in the C position:

dat Fred Susan Dafna haar pap **helpen laten eten** zal.
----- *zal*$_n$ Fred Susan Dafna haar pap **helpen laten eten** e$_n$

The central facts are now fact 4 and fact 5.

Fact 4: The verb cluster is a constituent, a complex verb (category V).

This is shown by the fact that the verb cluster itself can occur in first position, but its parts cannot:

(15) a. **Helpen laten eten** *zal* Fred Susan Dafna haar pap.
b. ***Helpen** *zal* Fred Susan Dafna haar pap **laten eten**.
c. * **Laten** *zal* Fred Susan Dafna haar pap **helpen eten**.
d. * **Eten** *zal* Fred Susan Dafna haar pap **helpen laten**.
e. ***Helpen laten** *zal* Fred Susan Dafna haar pap **eten**.
f. ***Helpen eten** *zal* Fred Susan Dafna haar pap **laten**.
g. * **Laten eten** *zal* Fred Susan Dafna haar pap **helpen**.

The facts 1-4 are explained if the verb cluster counts as a **single verb**: a complex which counts as a single lexical unit for the syntax. We call this a **serial verb**.

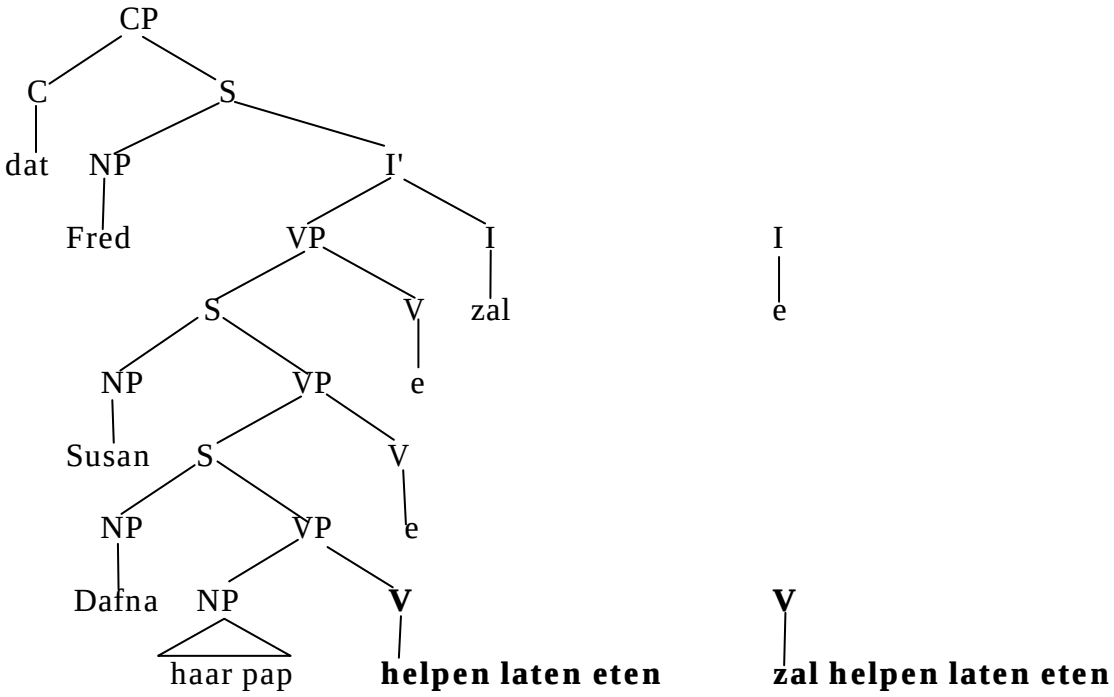
Fact 5: The verb cluster is sitting in the position of the **lowest V node** in the tree (and not, for instance, extraposed higher up on the right side of the tree, as earlier analyses have it).

This is shown by the fact that **haar pap helpen laten eten** forms a constituent for the verb second construction: crucially, the cases in (16a-c) are felicitous:

(16) a. **Haar pap helpen laten eten** *zal* Fred Susan Dafna.
b. **Dafna haar pap helpen laten eten** *zal* Fred Susan.
c. **Susan Dafna haar pap helpen laten eten** *zal* Fred
d. ***Fred Susan Dafna haar pap helpen laten eten** *zal*.

(16d) is out for the same reason as before, it moves the syntactic variable in I too high up in the tree.

This is evidence for the structure I gave above:



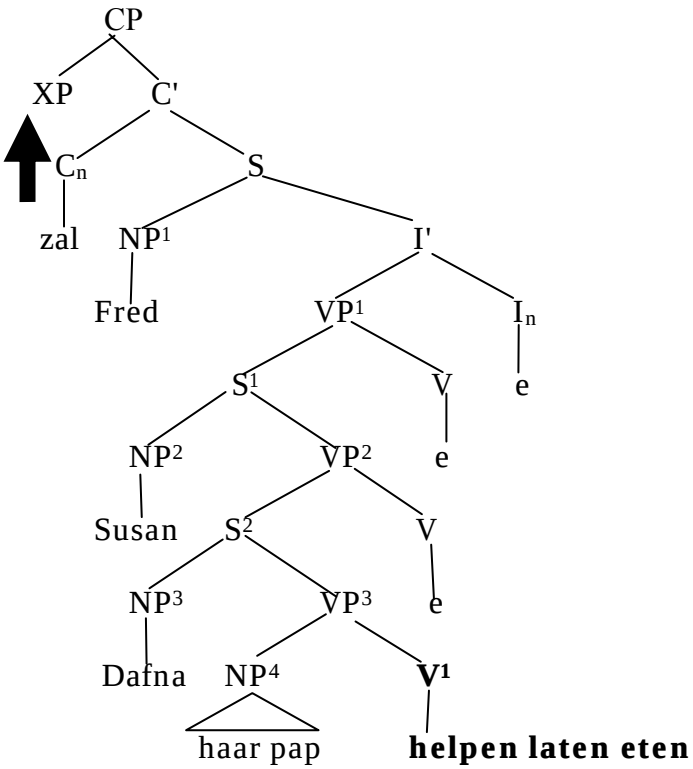
... dat Fred [Susan [Dafna [haar pap **helpen laten eten**] e] e] **zal**
that Fred Susan Dafna her porridge **help let eat** will

Incorporation:
Incorporation raids the verbal right side of the tree in Dutch, and incorporates all the verbs into a complex item of lexical category V (a serial verb).

-I assume that optionally even the tensed auxiliary can incorporate, giving the other order: **zal helpen laten eten**.
(Which is, in my dialect, much the preferred order.)

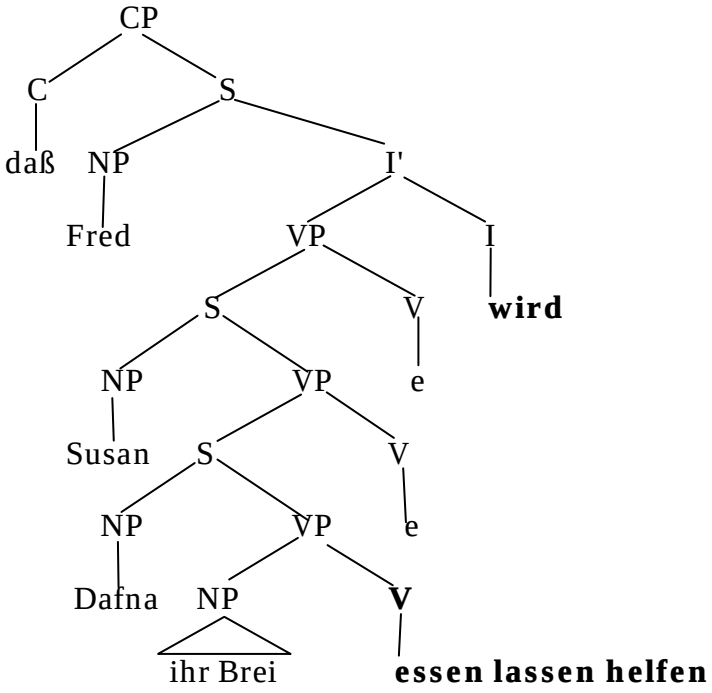
The verb second data shown above are explained by the verb second tree below:
All and only the subtrees NP^1, NP^2, NP^3, NP^4, VP^1, VP^2, VP^3, S^1, S^2 (you can't tell those appart from VP^1 and VP^2), and V can occur in position XP.

-Verb-second:



XP **zal$_n$** Fred [Susan [Dafna [haar pap **laten helpen eten**] e] e] e$_n$

The constituency arguments apply to German as well. Thus, the only difference between Dutch and German is **the order of infinitives inside the serial verb:**



INTERMEZZO:
A COMPUTATIONAL ASIDE ON GENERATIVE POWER

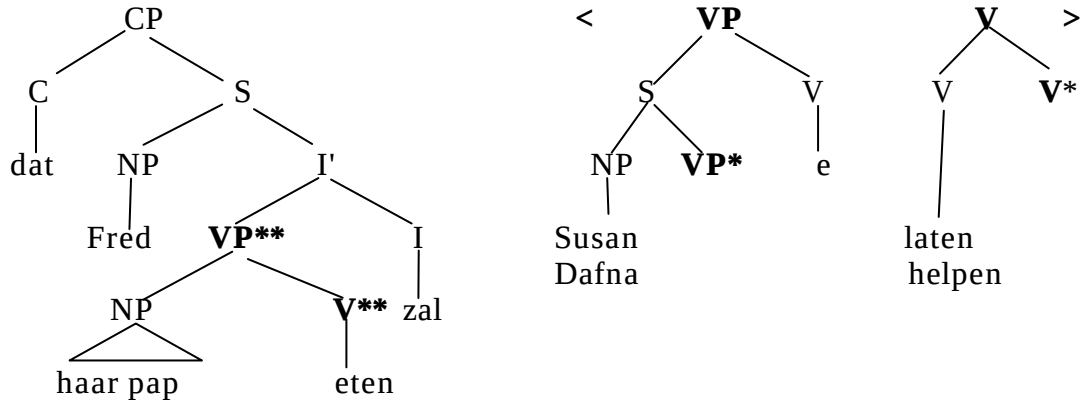
Assume, for the sake of discussion, an internal tree structure to the serial verb as a binary tree of verbs, say, [$_V$ [$_V$ helpen [$_V$ [$_V$ laten] [$_V$ eten]]]]. Think about the the tree set of Dutch, assuming the given analysis of the verb cluster.

-Tree set is is not context free.
-Tree set is not in the capacity of Aravind Joshi's Tree Adjoining Grammars.

(The analysis of Joshi 1983, 1987 does not account for the facts:
-the verb cluster is not a constituent,
-the verbs are sitting in the wrong place.)

The tree set is Mildly Contect-sensitive.
(semi linear grammars that allow polynomial parsing and cross-serial dependencies),
Multi-component Tree-adjoining Grammar (Weir 1987)
(Multi-adjoin complex trees into simple trees):

Multi-component TAG:



Adjoin simultaneously [$_{VP}$ [Dafna **VP**] e] in [$_{VP}$ [Susan **VP**] e]
and [$_V$ helpen **V**] in [$_V$ laten **V**]

which gives:

< [$_{VP}$ [Susan [Dafna **VP**] e]e] , [$_V$ laten [helpen **V**]] >

This adjoins simultaneously at the marked VP-site and the V-site in the VP.

SERIAL VERBS

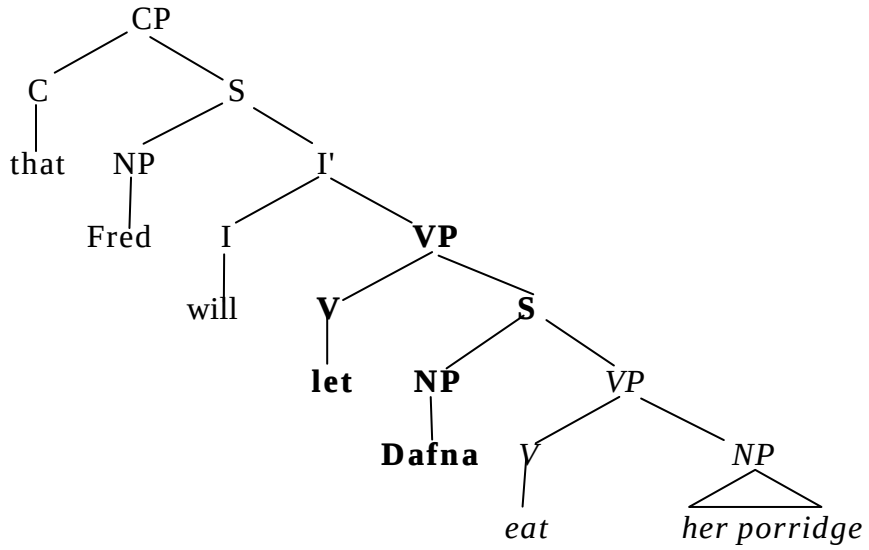
Syntactic complexes are turned into lexical items, which are treated (by and large) as lexical primitives by the syntax (like simple lexical items or the complex lexical items that are built by the lexicon).

Serial verbs are found in various African languages (for instance, Yoruba), and Asian languages (for instance Chinese), but it should be stressed that the term stands for syntacticaly and semantically distinct phenomena.

Most importantly:
Chinese and African serial verbs:
what are syntactically verb phrases are turned into lexical items.
The complexes that behave like units contain verbs **and their arguments**.

This means that no argument can be made for them for the existence of mechanims for creating n-place predicates.
For our purposes, the germanic construction is interesting precisely because the verbal unit does **not** include the arguments.

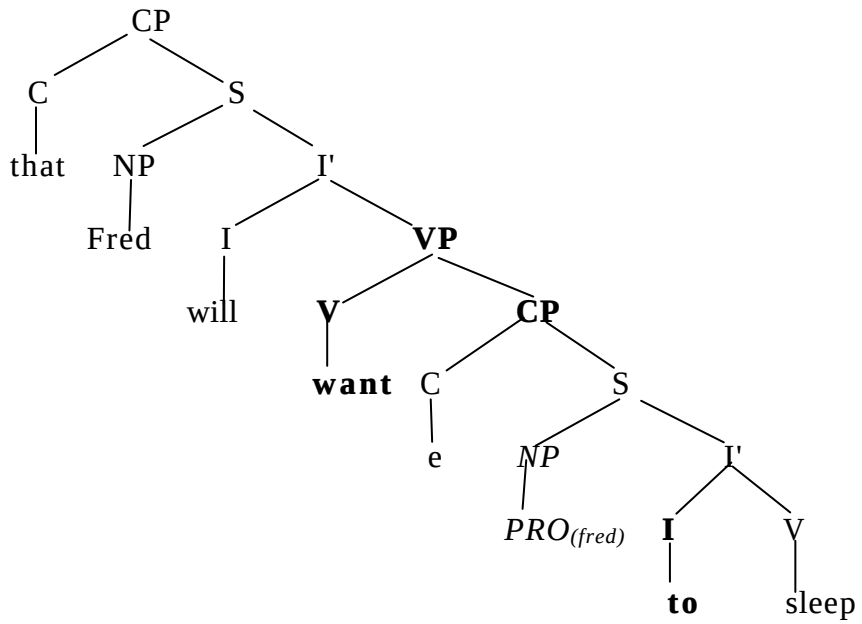
3. SYNTAX-SEMANTICS MISMATCHES IN SMALL CLAUSES



(17) that Fred will **let** *Dafna eat her porridge*.

Standard syntactic account:
- *Let, help….*: take a **small clause** complement (S)
- Small clause subject (*Dafna*) receives special, accusative case marking (dative, in the case of *help*).

-Similarly, standard: **infinitives** with marker ***to*** are **clausal** constructions, often with a empty pronominal subject called PRO:



(18) That Fred will **want** *to sleep*.

Standard claim (made by syntacticians) about the semantics of these constructions: (e.g. Higginbotham)

> **Clausal structures are interpreted as semantically saturated entities, propositions.**
>
> **e.g:** [$_S$ *Dafna eat her porridge*] = EAT(DAFNA, HER PORRIDGE)
> [$_{CP}$ PRO$_{fred}$ to sleep] = ^SLEEP(x) where x $\rightarrow$ Fred

Many semanticists argue for for a different semantics of infinitives:

> **Infinitives are interpreted as semantically unsaturated entities, properties.**
> (Intuition: *Fred wants to sleep* expressesthe relation that hold between Fred and the property SLEEPING, if Fred desires SLEEPING to be one of his propertes.)

And the standard claim (made by semanticists) about the syntax is that infinitives are not clausal structures (e.g. Dowty, Chierchia).

There are syntactic arguments in favor of the clausal analysis,
and semantic arguments in favor of the property analysis.

Landman 2003 argues against:
The Assumption of Perfect Matching between syntax and semantics.

Both syntacticians and semanticists have been guilty of using the APM to **insinuate** that you can do without the other:
-if you can read off the semantics from the syntax, or the "logical form" (as a level of syntax), you don't need semantics;
-if you make your semantic types fine-grained enough, you don't need any syntax.

Landman 2003:
Syntax and Semantics are at crucial points mismatched.
(Ray Turner called it **a Marriage of Inconvenience**)

Landman 2003:
syntax and semantics of indefinite noun phrases in argument position and in predicate position is mismatched.

My main proposal here is that the syntax and semantics of verbs like *let, help, see,…* is mismatched:
Clausal syntax + Non-propositional semantics

(19) Fred **lets** *Dafna eat her porridge*.

Perfect Matching: Clausal syntax, hence propositional semantics:
LET (Fred, *Dafna eat her porridge*),

where **LET** is a **two-place relation** between the interpretation of the subject *Fred* (the individual **Fred**) and the interpretation of the complement clause *Dafna eat her porridge* (a **proposition**).

Proposition: saturated entity (Frege 1894): in which the interpretation of *eat her porridge* is applied to the interpretation of *Dafna*.

-Not necessarily a set of worlds
-Landman 2000 (*Events and Plurality*): set of events (event type).

CLAUSAL ASSUMPTION + Perfect matching
LET is a **two-place relation** between an **individual** and a **proposition**.

In contrast, my assumption:

MISMATCH ASSUMPTION ONE

In the interpretation of the small clause [$_S$ *Dafna eat her porridge*],
the interpretation of the subject *Dafna* **(the individual Dafna)**
and the interpretation of the predicae *eat her porridge* **(a property)**
are not combined into a proposition.

[$_S$ *Dafna eat her porridge*] → **Dafna** + λx. **EAT**(x, **Por**)

(I use **Por** for Dafna's porridge, ignoring the pronoun completely)
Thus, the small clause is not interpreted as a saturated entity.

Semantics of *let, help, see, …* (and control verbs as well, for that matter):

MISMATCH ASSUMPTION TWO

LET is a **three-place relation** between **two individuals** and a **property**

LET is a relation between the interpretation of **the external subject** (x),
the interpretation of **the small clause subject** (y),
and the interpretation of **the small clause predicate** (P):

LET → λPλyλx. **LET**(x, P(y))
(x let's y have property P.)

Ultimately, the clausal complement **will** express a proposition,but only **by the lexical meaning of LET**.

There is no level of grammatical (as opposed to lexical) derivation, where the small class expresses a proposition.
This is exactly what I mean by **mismatch**.

LET → **λPλyλx. LET (x, P(y))**
HELP → **λPλyλx. HELP(x, P(y))**
SEE → **λPλyλx. SEE (x, P(y))**
HEAR → **λPλyλx. HEAR(x, P(y))**

4. SEMANTICS OF SERIAL VERBS: FUNCTION COMPOSITION

The next semantic assumption is a standard one.

-In **phrasal domains** (IP (S), VP), the basic **meaning composition** operation is:

> **Function-argument application:**
> **APPLY[FUNCTION f, ARGUMENT a] = (f(a))**

-In **lexical domains** (like V), the basic **meaning composition** operation is:

> **Function composition:**
> **COMPOSE[FUNCTION f, FUNCTION g] = f ° g**
> **where f ° g = λx. f(g(x))**

f ° g is the function that takes its inputs in the domain of g and has its outputs in the range of f, and assigns to every x in the domain of g, the result of applying first g to x, (= g(x)), and then f to the result f(g(x)).

The formula for composition:

1. Apply g to a variable x: g(x)
2. Check that g(x) is of the right type to be the argument of f.
3. Apply f to g(x)
4. Abstract over variable x.

Actually, the operation needed in semantics is a bit more general:

> **GENERALIZED COMPOSITION:**
> **f ° g = λx_n…x_1. f(g(x_1,…,x_n))**

The formula for generalized composition:

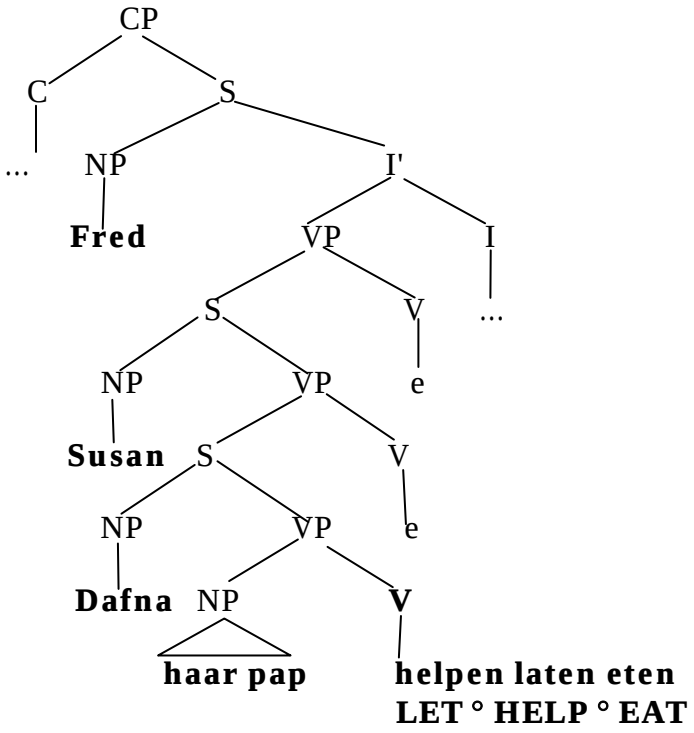
1. Apply g to a variable x_1.,,,x_n: g(x_1,…,x_n)
2. Untill g(x_1,…,x_n) is of the right type to be the argument of f.
3. Apply f to g(x_1,…,x_n)
4. Abstract over variables x_1,…,x_n.

(Quite a bit of literature on function composition in lexical domains, e.g. Hoeksema 1984.)

COMPOSITION ASSUMPTION:
The verbs in the serial verb cluster combine through generalized composition.

Claim: The mismatch assumption and the composition assumption are the only assumptions that need to be made to get the semantics of the serial verb cluster come out right.

Example: (ignoring the modal *zal*)



Predicted Semantics:

LET = λPλyλx. **LET**(x, P(y)) **EAT** = λzλy. **EAT**(y,z)

LET ° EAT = λPλyλx. **LET**(x, P(y)) ° λzλy. **EAT**(y,z) =

λz **[**λPλyλx. **LET**(x, P(y))**] (** λzλxy. **EAT**(y,**z**) (z) **)** =
λz **[**λ**P**λyλx. **LET**(x, **P**(y))**] (**λy. **EAT**(y,z) **)** =
λz **[** λyλx. **LET**(x, λ**y**. **EAT**(**y**,z) (y))**]** =

λzλyλx. **LET**(x, **EAT**(y,z)) *x lets*: *y eat z*

Conclusion:			
2 place verb	**+ LET**	→	**3-place verb**
λzλy. **EAT**(y,z)	**LET**		λzλyλx. **LET**(x, **EAT**(y,z))
1 2			**1 2 3**

Next:
HELP ° (LET ° EAT) =

λPλxλu. **HELP**(u, P(x)) ° λzλyλx. **LET**(x, **EAT**(y,z)) =

λzλy **[**λPλxλu. **HELP**(u, P(x))**] (**λ**zλy**λx. **LET**(x, **EAT**(**y,z**)) (y,z)) =
λzλy [λ**P**λxλu. **HELP**(u, **P**(x))**] (**λx. **LET**(x, **EAT**(y,z))) =

λzλyλxλu. **HELP**(u, **LET**(x, **EAT**(y,z)))

3 place verb	**+ HELP→**	**4-place verb**
λzλyλx. **LET**(x, **EAT**(y,z))		λzλyλxλu. **HELP**(u, **LET**(x, **EAT**(y,z)))
1 2 3		**1 2 3 4**

So: **helpen laten eten** → λzλyλxλu. **HELP**(u, **LET**(x, **EAT**(y,z)))
u helps x; x lets y, y eats z.

-The basic composition operation in phrasal domains is **function-argument application:**
Type theory:
The left-rightorder in the λ-prefix represents the order of application.

-The meaning of the V ***helpen laten eten*** applies to the meaning of ***haar pap*** (λz)
-the result applies to the meaning of ***Dafna,*** (λy),
-the result to the meaning of ***Susan*** (λx),
-the result to the meaning of ***Fred*** (λu),
giving, the correct meaning for the sentence:

APPLY[, Fred]
APPLY[, Susan]
APPLY[, Dafna]
APPLY[HELP ° (LET ° EAT), Her Porridge]

=

HELP(Fred, LET(Susan, EAT(Dafna,Her Porridge)))

The semantics proposed gets the meanings right within a framework of standard assumptions about the semantic composition operations applicable in different domains (composition and application).

Given the meanings of the verbs that enter into the serial verb, composition has the effect of:

> **n-PLACE SERIAL VERB FORMATION:**
>
> **Let α be one of LET, HELP, SEE, HEAR ,…**
> **Let β be an n−1 place relation, then α ° β is an n-place relation.**

INTERMEZZO:
A COMPUTATIONAL ASIDE ON SEMANTIC PARSING

Dat Fred Susan Dafna haar pap helpen laten eten (zal)
Daß Fred Susan Dafna ihr Brei essen lassen helfen (wird)

Semantic parsing: Use type theory to find the interpretation:

Look for sentence meaning φ
We are given individual: *Fred*
Type theory resolves: $\varphi = \mathbf{P^1}(\text{Fred})$
Look for one-place predicate meaning $\mathbf{P^1}$, etc.

Dat/Daβ:	φ	
Fred:	$\mathbf{P^1}$(f)	$\varphi = \mathbf{P^1}(f)$
Susan:	$\mathbf{P^2}$(f,s)	$\mathbf{P^1} = \mathbf{P^2}(s)$ (and $\mathbf{P^2}(s))(f) = \mathbf{P^2}(f,s)$)
Dafna	$\mathbf{P^3}$(f,s,d)	$\mathbf{P^2} = \mathbf{P^3}(d)$
haar pap/ ihn brei	$\mathbf{P^4}$(f,s,d,p)	$\mathbf{P^3} = \mathbf{P^4}(p)$
Infinitive:	$(\boldsymbol{Z} \circ \mathbf{P^4})$(f,s,d,p)	*Z* for the contribution of the tensed modal in I

Dutch: **Forward parsing:**

helpen	(***Z*** ° **(HELP ° P^3**)(f,s,d,p)	**P^4** = **HELP** ° **P^3**
laten	(***Z*** ° **HELP ° (LET ° P^2**)(f,s,d,p)	**P^3** = **LET** ° **P^2**
eten	(***Z*** ° **HELP ° LET ° EAT**)(f,s,d,p)	**P^2** = **EAT**
zal	(**WILL ° HELP ° LET ° EAT**)(f,s,d,p)	*Z* = **WILL**
done		

German: **Store:**

essen	(***Z*** ° **P^4**)(f,s,d,p)	+ (**EAT**)^2
lassen	(***Z*** ° **P^4**)(f,s,d,p)	+ (**LET ° EAT**)^3
helfen	(***Z*** ° **P^4**)(f,s,d,p)	+ (**HELP ° LET ° EAT**)^4
	Retrieve:	
	(***Z*** ° (**HELP ° LET ° EAT**))(f,s,d,p)	
wird	(**WILL ° HELP ° (LET ° EAT**)(f,s,d,p)	
done		

There is a direct parsing strategy for German, but it involves search variables of complex types:

German: **Direct parsing with Complex variables:**

essen	(*Z* ° (**R^3** ° **EAT**))(f,s,d,p)	type: **R^3**(x,y,P)
lassen	(*Z* ° ((**R^2** ° **LET**) ° **EAT**))(f,s,d,p)	type: **R^2**(x,P)
helfen	(*Z* ° (**HELP ° LET ° EAT**))(f,s,d,p)	
wird	(**WILL ° HELP ° (LET ° EAT**)(f,s,d,p)	
done		

In either case, the German strategy is more complex than the Dutch strategy.

Bach,Brown and Marslen-Wilson, 1986:
Cross-linguistic experiment, showing that Dutch speakers parse Dutch serial verb constructions faster than German speakers parse analogous German serial verb constructions.

5. COMPLEXITY OF n-PLACE RELATIONS

The *let*, *help, see*,.. obviously provide a productive way of producing **3-place relations**:

(20) Dat Susan Dafna haar pap zal **laten eten**.
Susan will let Dafna eat her porridge.

4-place relations occur quite frequently:

(21) Dat Fred Susan Dafna haar pap zal **helpen laten eten**
Fred will help Susan let Dafna eat her porridge

(Anecdotal evidence:
-Dafna, age 10, rejected (21), when asked, and spontaneously used another 4-place case in the same week.
-The construction is found in literature (e.g. Mulisch).)

Note: alternative with the same meaning that does **not** require a semantic 4-place relation: *te-infinitive extraposition*

(22) Dat Fred Susan zal *helpen* [**Dafna haar pap te laten eten**]
Fred will help Susan to let Dafna eat her porridge

(22) is easier to process than (21).

5-place relations become hard to process:

(23) ?Dat Nirit Fred Susan Dafna haar pap zal **zien helpen laten eten**.
Nirit will see Fred help Susan let Dafna eat her porridge.

This is very hard, unlike the equivalent extraposed version, which is much simpler

(24) Dat Nirit Fred Susan zal **zien helpen** ***Dafna haar pap te laten eten.***
Nirit will see Fred help Susan to let Dafna eat her porridge

6-place relations: even the extraposed cases become inpenetrable:

(25) ?Dat Edit Nirit Fred Susan Dafna haar pap zal **laten zien helpen laten eten,**
Edit will let Nirit See Fred help Susan let Dafna eat her porridge.

(26)?Dat Edit Nirit Fred Susan zal **laten zien helpen** *Dafna haar pap te laten eten*
Edit will let Nirit see Fred help Susan to let Dafna eat her porridge

Note that the complexity is not related to the number of verbs involved, but the arity of the relation.

In Dutch, modal *moeten* (*must*) and also *willen* (w*ant*) takes an infinitive without *te* and are unproblematically part of the serial verb construction: they add modality, intensionality, but not an extra argument.

(21) Dat Fred Susan Dafna haar pap zal **helpen laten eten**
Fred will help Susan let Dafna eat her porridge

(27) Dat Fred Susan Dafna haar pap zal **moeten** helpen laten eten
Fred **must** (future) help Susan let Dafna eat her porridge.

(28) Dat Fred Susan Dafna haar pap zal **willen** helpen laten eten
Fred will **want** to help Susan let Dafna eat her porridge.

(29) Dat Fred Susan Dafna haar pap zal **moeten willen** helpen laten eten
Fred **must** (future) **want** to help Susan let Dafna eat her porridge.

(27)-(29): -**more verbs in the sequence** than in (21)
-same arity: 4-place predicates

And the cases in (27-(29) are not more difficult to process than (21), considerably easier than the 5-place and 5-place relations in (23) and (25).

Conclusion:

The complexity problems come in with n-place relations.
There is a reason why natural languages do not have lexical 5-or-more -place relations:
when the language has a mechanism for productively building 5-or-more-place relations, it doesn't process them well.

WHITHER n-PLACE RELATIONS IN SEMANTICS?

1. Verbal polyadicity.

e.g. McConnel-Ginet 1982: argument extension theory:

(30) a.	Mary **ate.**	$\mathbf{EAT}^1$(m)
b.	Mary **ate** a sandwich	$\mathbf{EAT}^2$(m,s)
c.	Mary **ate** a sandwich with a knife	$\mathbf{EAT}^3$(m,s,k)
d.	Mary **ate** a sandwich with a knife at midnight	$\mathbf{EAT}^4$(m,s,k,m)

Two issues:

1. Argument drop:

(30a) Mary **ate**.

polyadicity vs. argument drop:

$\mathbf{EAT}^1$(m)

$\exists$x[$\mathbf{EAT}^2$(m,x)]

2. Arguments vs. Adjuncts:

(30c) Mary **ate** a sandwich with a knife

$\mathbf{EAT}^3$(**m**,**s**,**k**)

3 arguments

$\exists$e[$\mathbf{EAT}^3$(e,**m**,**s**) $\wedge$ **Instrument**(e)=**k**] Davidsonian

2 arguments, one adjunct, 1 implicit argument (e)

$\exists$r[$\mathbf{EAT}^1$(e) $\wedge$ **Agent**(e)=**m** $\wedge$ **Theme**(e)=**s** $\wedge$ **Instrument**(e)=**k**]

Neo-davidsonian

Central fact:

Adding adverbial modifiers, prepositional phrases does not lead to processing difficulties of adicity.

Central moral:

-Adicity is grammatically specified (and restricted to 4 or less).

-Implicit parameters, time, place, world, do not contribute to adicity

-Semantics of adverbials, prepositional phrases:

semantic adjunction: functions from event predicates to event predicates.

Adjuncts do not contribute to adicity.

-Against semantic argument-addition theories.

-Reason to carefully scrutinize theories of hierarchical adverbial functional projections (Chinque's 57 varieties):

The **reason** adverbial modifiers don't lead to processing difficulties is the **flat semantics** of adjunction.

-Subcategorisation vs. hierarchy. Subcategorisation doesn't cost much,

-Compley hierarchy (like adicity) is encoding of numbers which need to be kept track of.

2. n-ary quantification of scopelessness.

Scha 1984, also, Hintikka, Barwise, van Benthem 1986 , May 1986, Sher.

(31) 5000 firemen put out 30 fires in 7 states with 900 firetrucks

Cumulative readings: **scopeless readings:**

The total number of firemen that put out fires in some state is 5000
The total number of fires put out by firemen in some state is 30
The total number of states where firemen put our fires is 7
The total number of firetrucks used by firemen to put out fires in some state is 900.

Scha (and others): semantics with **n-place relations:**

PUT OUT-IN-WITH^{4}(5000 firemen,30 fires,7 states, 900 firetrucks)

Landman 2000, Schwarzschild and others:
event semantics plus semantics of plurality deals with cumulative readings without adding to adicity.

3. Relational scope mechanism.

Scope ambiguities:

(32) a. A bus is waiting at the finish for all participants who finish the race. $\exists\forall$
b. A medal is waiting at the finish for all participants who finish the race $\forall\exists$

Standard scope mechanism: clausal mechanism:
Frege 1879 (preface), Montague, Lakoff, May, etc.:
Quantifier raising, lowering, substitution,…
Operates on (the interpretation of) a noun phrase in a clause:
-the noun phrase is interpeted as a variable *in situ*
-the variable is bound higher up, where the noun phrase is interpreted.

Hendriks 1987: Relational mechanism:

A relation between individuals shifts to different relations between generalized quantifiers, showing different scope orders:

Two-place:

$\lambda y\ \lambda x.$	$\mathbf{R^2}(x,y)$	**shifts to**
$\lambda\mathbf{NP_2}\,\lambda\mathbf{NP_1}.$	$(\mathbf{NP_1}(\lambda x.\mathbf{NP_2}(\lambda y.\mathbf{R^2}(x,y)))$	Direct scope
$\lambda\mathbf{NP_2}\,\lambda\mathbf{NP_1}.$	$(\mathbf{NP_2}(\lambda x.\mathbf{NP_1}(\lambda y.\mathbf{R^2}(x,y)))$	Indirect scope

Advantages: -more **local, bounded** mechanism (relation is required)
Disadvantages: -n-place relations are required.

(33) (as the photographs show:)
John kissed Mary **on** *some summer day* **in** *every park*

Relational mechanism: noun phrases gtet scope by:
shifting the interpretation of the co-argument places of a relation.

In (33) two scopal noun phrases are inside adjuncts:
The scope mechanism requires a **4-place relation**: **KISS-ON-IN**^{4} to give these two arguments different scopes with respect to each other.

The same situation Joshi, Kallmeyer, Romero 2006's theory of scope through Multi constituent adjunction:
They need to form an extended lexical tree into which the arguments of the verb and of the prepositions are adjoined, with different scopal orders. This is an n-place relation.

Processing questions for scope ambiguities: inverse scope is often hard to get.
But these interpretation problems exist already for two-place relations.
It is not clear that there is a relation between this and the adicity problem.

4. Discourse representation theory, dynamic semantics.

(34) a. *A farmer kept a donkey in a stable*. He took care of it.
b. If a farmer kept a donkey in a stable, he took care of it.

$$\lambda\varphi.\,\exists[\ \boldsymbol{\lambda x\lambda y\lambda z\lambda t\lambda e.}\ \text{FARMER}(x) \wedge \text{DONKEY}(y) \wedge \text{STABLE}(z) \wedge \text{PAST}(t) \wedge \text{TIME}(e)=t \wedge \text{KEEP}(e,x,y,z) \wedge \lambda x\lambda y\lambda z\lambda t\lambda e.\varphi\,(x,y,z,t,e)\ ]$$

It is quite fruitful to think of information built up in discourse to be **relational:**
i.e. the five-place relation in the interpretation of (34).
Also think of interpretation and domain of verification:

(35) a. Every boy loves a girl.
b. $\forall$[BOY, λx.$\exists$[GIRL, λy.LOVE(x,y)]]
c. $\lambda x\lambda y$. BOY(x) $\wedge$ GIRL(y) $\wedge$ LOVE(x,y) **2-place discourse relation**

We **evaluate** interpretation (35b), by checking the extension of relation (35c). In general, this requires n-plece relations.

Suggestion:
-n-place relations are used unrestrictedly in dynamic semantics of discourse.
-There is a difference between discourse semantics and compositional semantics (grammar).
-n-place relations enter into compositional semantics only cautiously
(Landman, *Two Tier Semantics,* in progress)