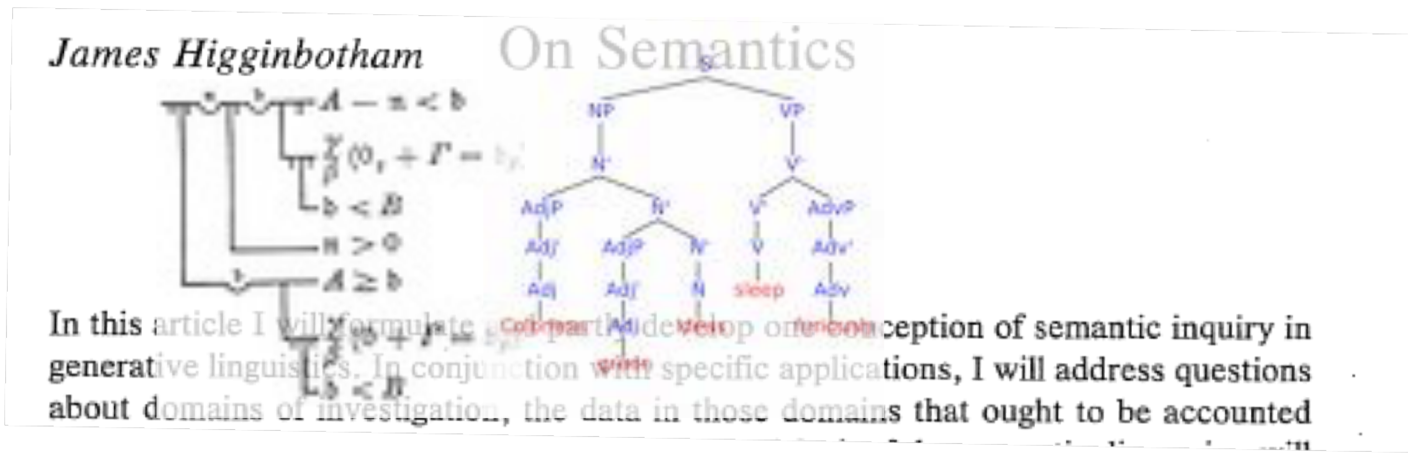
Locating Human Meanings:

Less Typology, More Constraint



Paul M. Pietroski, University of Maryland

Dept. of Linguistics, Dept. of Philosophy

Elizabeth, on her side, had much to do. She wanted to ascertain the feelings of each of her visitors, she wanted to compose her own, and to make herself agreeable to all; and in the latter object, where she feared most to fail, she was most sure of success, for those to whom she endeavoured to give pleasure were prepossessed in her favour.

Bingley was ready,
Georgiana was eager, and
Darcy determined to be pleased.

Jane Austen
Pride and Predjudice

Bingley is eager to please.

(a) Bingley is eager to be *one who pleases*.

#(b) Bingley is eager to be *one who is pleased*.

Bingley is easy to please.

#(a) Bingley can easily *please*.

(b) Bingley can easily *be pleased*.

Human children naturally acquire languages
that somehow generate boundlessly many expressions
that connect meanings (whatever they are)
with pronunciations (whatever they are)
in accord with certain constraints.

Human languages generate boundlessly many expressions
that connect meanings with pronunciations
in accord with certain constraints.

Do human linguistic expressions exhibit meanings of different *types*?

(1) Fido	(5) every cat
(2) chase	(6) chase every cat
(3) every	(7) Fido chase every cat
(4) cat	(8) Fido chased every cat.

And if so, *which meaning types* do they exhibit?

What are the Human Meaning Types?

- one familiar answer, via Frege's conception of *ideal* languages

  (i) a basic type <e>, for *entity denoters*

  (ii) a basic type <t>, for *thoughts* or *truth-value denoters*

  (iii) if <α> and <β> are types, then so is <α, β>

Fido, Garfield, Zero, …

Fido barked.
Fido chased Garfield.
Zero precedes every positive integer.

What are the Human Meaning Types?

- one familiar answer, via Frege's conception of ***ideal*** languages

    (i) a basic type <e>, for *entity denoters*

    (ii) a basic type <t>, for *thoughts* or *truth-value denoters*

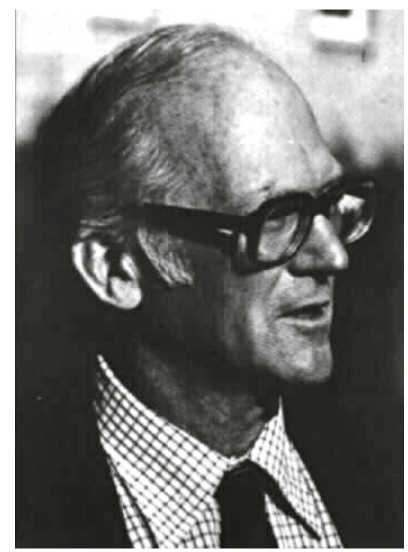
    (iii) if <α> and <β> are types, then so is <α, β>

- on the other hand, one might suspect

    (a) there are no *meanings* of type <e>

    (b) there are no *meanings* of type <t>

    (c) the recursive principle is ~~crazy~~ implausible



What are the Human Meaning Types?

- one familiar answer, via Frege's conception of *ideal* languages

  (i) a basic type <e>, for *entity denoters*

  (ii) a basic type <t>, for *thoughts* or *truth-value denoters*

  (iii) if <α> and <β> are types, then so is <α, β>

  That's a lot of types

a basic type <e>, for entity denoters

a basic type <t>, for truth-value denoters

if <α> and <β> are types, then so is <α, β>

at Level 5,
more than 5 x 10^{12}

---

0. <e> <t> — (2) types at Level Zero

1. <e, e> <e, t> <t, e> <t, t> — (4) at Level One, all <0, 0>

2. eight of <0, 1> eight of <1, 0>
   sixteen of <1, 1>

   (32), including <e, et> and <et, t>

3. 64 of <0, 2> 64 of <2, 0>
   128 of <1, 2> 128 of <2, 1>
   1024 of <2, 2>

   (1408), including <e, <e, et>>; <et, <et, t>>; and <<e, et>, t>

4. 2816 of <0, 3> 2816 of <3, 0>
   5632 of <1, 3> 5632 of <1, 3>
   45,056 of <2, 3> 45,056 of <3, 2>
   1,982,464 of <3, 3>

   (2,089,472), including <e, <e, <e, <et>> and <<e, et>, <<e, et>, t>

a basic type <e>, for entity denoters
a basic type <t>, for truth-value denoters
if <α> and <β> are types, then so is <α, β>

---

0. <e> <t> ziggy

Number(ziggy)

1. <e, t> λx.Number(x)

2. <e, et> λy.λx.Predecessor(x, y)

λy.λx.Precedes(x, y)

3. <<e, et>, t> Transitive[λy.λx.Precedes(x, y)]

Intransitive[λy.λx.Predecessor(x, y)]

4. <<e, et>, <<e, et>, t>

TransitiveClosure[λy.λx.Precedes(x, y), λy.λx.Predecessor(x, y)]

Frege *invented* a language that supported abstraction on *relations*

Three precedes four.

Three is something *that precedes four*. λx.Precedes(x, 4)

Four is something *that three precedes*. λx.Precedes(3, x)

*Precedes is somerelat *that three four*. λ**R**.**R**(3, 4)

The plate outweighs the knife.

The plate is something *which* *outweighs the knife*.

The knife is something *which the plate outweighs* .

*Outweighs is somerelat *which the plate* *the knife*.

a basic type <e>, for entity denoters

a basic type <t>, for truth-value denoters

if <α> and <β> are types, then so is <α, β>

…

3. <<e, et>, t> Transitive[λy.λx.Precedes(x, y)]

Precedes transits.

4. <<e, et>, <<e, et>, t>

TransitiveClosure[λy.λx.Precedes(x, y), λy.λx.Predecessor(x, y)]

Precedes transits predecessor.

a basic type <e>, for entity denoters

a basic type <t>, for truth-value denoters

if <α> and <β> are types, then so is <α, β>

---

0. <e> <t> — (2) types at Level Zero

1. <e, e> <e, t> <t, e> <t, t> — (4) at Level One, all <0, 0>

2. eight of <0, 1> eight of <1, 0>
sixteen of <1, 1>
— (32), including <e, et> and <et, t>

3. 64 of <0, 2> 64 of <2, 0>
128 of <1, 2> 128 of <2, 1>
1024 of <2, 2>
— (1408), including <e, <e, et>>; <et, <et, t>>; and <<e, et>, t>

4. 2816 of <0, 3> 2816 of <3, 0>
5632 of <1, 3> 5632 of <1, 3>
45,056 of <2, 3> 45,056 of <3, 2>
1,982,464 of <3, 3>
— (2,089,472), including <e, <e, <e, <et>> and <<e, et>, <<e, et>, t>

What are the Human Meaning Types?

- one familiar answer, via Frege's conception of *ideal* languages

  (i) a basic type <e>, for *entity denoters*

  (ii) a basic type <t>, for *thoughts* or *truth-value denoters*

  (iii) if <α> and <β> are types, then so is <α, β>

- a suggestion in the footnotes of "On Semantics"

  Filter Functionals:

  no <α, β> types where α is *non-basic*

  <et, t>   <e, <e, <e, <e, t>>>

What are the Human Meaning Types?

- one familiar answer, via Frege's conception of *ideal* languages

  (i) a basic type <e>, for *entity denoters*

  (ii) a basic type <t>, for *thoughts* or *truth-value denoters*

  (iii) if <α> and <β> are types, then so is <α, β>

- a suggestion less permissive than "*Filter Functionals*"

  No Recursion: no <α, β> types

  (1) a basic type <M>, for *monadic predicates*

  (2) a basic type <D>, for *dyadic predicates*

  ...

  (*n*) a basic type <N>, for *N-adic predicates*

What are the Human Meaning Types?

- one familiar answer, via Frege's conception of ***ideal*** languages

  (i) a basic type <e>, for *entity denoters*

  (ii) a basic type <t>, for *thoughts* or *truth-value denoters*

  (iii) if <α> and <β> are types, then so is <α, β>

- a suggestion much less permissive than "*Filter Functionals*"

  No Recursion: no <α, β> types

  (1) a basic type <M>, for *monadic predicates*

  (2) a basic type <D>, for *dyadic predicates*

  Minimal Relationality

Degrees of "Semantic Relationality"

- None: *e.g.*, Monadic Predicate Calculi
  - some M is (also) P

- Unbounded: *e.g.*, Tarski-style Predicate Calculi
  - Mx & Py & Syz & Rxw & Bzuv & …

a Tarski-style Predicate Calculus permits Unbounded Adicity

Brown(x)	1
Brown(x) & Dog(x)	1
Saw(x, y)	2
Dog(x) & Saw(x, y)	2
Dog(x) & Saw(x, y) & Cat(z)	3
Dog(x) & Saw(x, y) & Cat(z) & Saw(z, w)	4
Dog(Fido) & Saw(Fido, Garfield)	0
Between(x, y, z)	3
Quartet(x, y, z, w)	4
Between(x, y, z) & Quartet(w, x, y, x)	4
Between(x, y, z) & Quartet(w, v, y, x)	5
Between(x, y, z) & Quartet(w, v, u, y)	6
Between(x, y, z) & Quartet(w, v, u, t)	7

unbounded adicity,
but no typology
…
each expression (wff)
is a <u>*sentence*</u>
…
and each <u>*sentence*</u>
is <u>*satisfied*</u> by
all/some/no
<u>*sequences*</u> of
domain entities

Degrees of "Semantic Relationality"

- None: *e.g.*, Monadic Predicate Calculi
  - some M is (also) P

- Some, but Less Than Unbounded
  - Minimally Relational (maximally limited)
  - "Mildly" Relational (severely limited)
  - Bounded, but still "pretty permissive"

- Unbounded: *e.g.*, Tarski-style Predicate Calculi
  - Mx & Py & Syz & Rxw & Bzuv & …

Plan for Rest of the Talk

- Characterize a notion of "Minimally Relational"
- Describe a Possible Language that is Minimally Relational and (correlatively) "Minimally Interesting" in this respect
- Suggest that while Human Meanings may be *a little* more interesting, they approximate Minimal Relationality
- End with reminders of some other respects in which Human Languages seem to be Minimally Interesting, and suggest that semantic typology is yet another case

Minimally Relational

- admit ***dyadic*** predicates, but no predicates of higher adicity
  - ABOVE(_, _) and CAUSE(_, _) are OK; so is AGENT(_, _)
  - SELL(_, _, _, _) and BETWEEN(_, _, _) are not-OK
- admit relational notions only in the *lexicon*
  - BETWEEN(_, _, JIM) is not-OK
  - ON(_, _) & HORSE(_) is not-OK
- correspondingly limited *combinatorial operations*
  - if ON(_, _) and HORSE(_) combine, the result is monadic
  - combining lexical items cannot yield relational notions

We can imagine a language whose expressions are limited to…

(1) finitely many *atomic monadic* predicates: M_1(_) … M_k(_)

(2) finitely many *atomic dyadic* predicates: D_1(_, _) … D_j(_, _)

(3) boundlessly many *complex monadic* predicates

Monad + Monad ➔ Monad

BROWN(_) + HORSE(_) ➔ BROWN(_)^HORSE(_)

FAST(_) + BROWN(_)^HORSE(_) ➔ FAST(_)^BROWN(_)^HORSE(_)

We can imagine a language whose expressions are limited to…

(1) finitely many *atomic monadic* predicates: $M_1(\_)$ … $M_k(\_)$

(2) finitely many *atomic dyadic* predicates: $D_1(\_, \_)$ … $D_j(\_, \_)$

(3) boundlessly many *complex monadic* predicates

Monad + Monad ➔ Monad

for each entity:

$\Phi(\_)\wedge\Psi(\_)$ applies to it

if and only if

$\Phi(\_)$ applies to it, *and*
$\Psi(\_)$ applies to it

We can imagine a language whose expressions are limited to…

(1) finitely many *atomic monadic* predicates: $M_1(\_)$ … $M_k(\_)$

(2) finitely many *atomic dyadic* predicates: $D_1(\_, \_)$ … $D_j(\_, \_)$

(3) boundlessly many *complex monadic* predicates

Monad + Monad ➔ Monad

Dyad + Monad ➔ Monad

for each entity:

$\Phi(\_)$^$\Psi(\_)$ applies to it

if and only if

$\Phi(\_)$ applies to it, *and*

$\Psi(\_)$ applies to it

ON(_, _) + HORSE(_)

🠻

∃[ON(_, _)^HORSE(_)]

|_______|__________|

(thing that is) on a horse

We can imagine a language whose expressions are limited to…

(1) finitely many *atomic monadic* predicates: M_1(_) … M_k(_)

(2) finitely many *atomic dyadic* predicates: D_1(_, _) … D_j(_, _)

(3) boundlessly many *complex monadic* predicates

Monad + Monad ➔ Monad

for each entity:

Φ(_)^Ψ(_) applies to it

if and only if

Φ(_) applies to it, *and*
Ψ(_) applies to it

Dyad + Monad ➔ Monad

ON(_, _) + HORSE(_)

🡻

∃[ON(_, _)^HORSE(_)]

(thing that is) on a horse
thing that a horse is on

We can imagine a language whose expressions are limited to…

(1) finitely many ***atomic monadic*** predicates: $M_1(\_)$ … $M_k(\_)$

(2) finitely many ***atomic dyadic*** predicates: $D_1(\_, \_)$ … $D_j(\_, \_)$

(3) boundlessly many ***complex monadic*** predicates

Monad + Monad ➔ Monad

for each entity:

$\Phi(\_)$^$\Psi(\_)$ applies to it

if and only if

$\Phi(\_)$ applies to it, *and*
$\Psi(\_)$ applies to it

Dyad + Monad ➔ Monad

for each entity:

$\exists[\Delta(\_, \_)$^$\Psi(\_)]$ applies to it

if and only if

it bears Δ to *something*
that $\Psi(\_)$ applies to

∃[AGENT(_, _)^HORSE(_)]^EAT(_)^FAST(_)

is like

∃e[AGENT(e’, e) & HORSE(e)] & EAT(e’) & FAST(e’)

∃[AGENT(_, _)^FAST(_)^HORSE(_)]^EAT(_)

is like

∃e[AGENT(e’, e) & FAST(e) & HORSE(e)] & EAT(e’)]

We don’t need variables to capture the meanings of ‘horse eat fast’ and ‘fast horse eat’.

SEE(_)^∃[THEME(_, _)^HORSE(_)]

is like

SEE(e') & ∃e[THEME(e', e) & HORSE(e)]

SEE(_)^∃[THEME(_, _)^∃[AGENT(_, _)^HORSE(_)]^EAT(_)]

is like

SEE(e'') & ∃e'[THEME(e'', e') & ∃e[AGENT(e', e)^HORSE(e)] & EAT(e')]

We don't need variables to capture the meanings of 'see a horse' and 'see a horse eat'.

What are the Human Meaning Types?

--two basic types, <e> and <t>

--endlessly many derived types of the form <α, β>

-- <α> can combine with <α, β> to form <β>

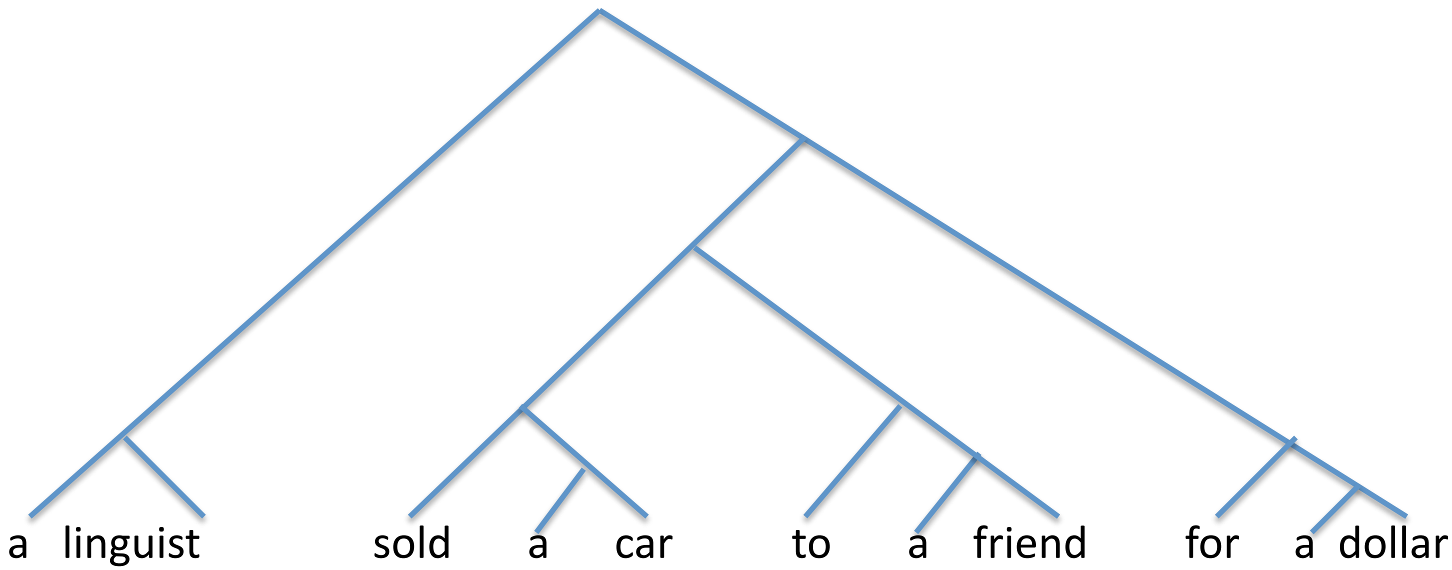
--a monadic type <M>

--a dyadic type <D>, for finitely many atomic expressions

-- <M> + <M> → <M>

<M> + <D> → <M>

Can Human Lexical Items have “Level Four Meanings”?

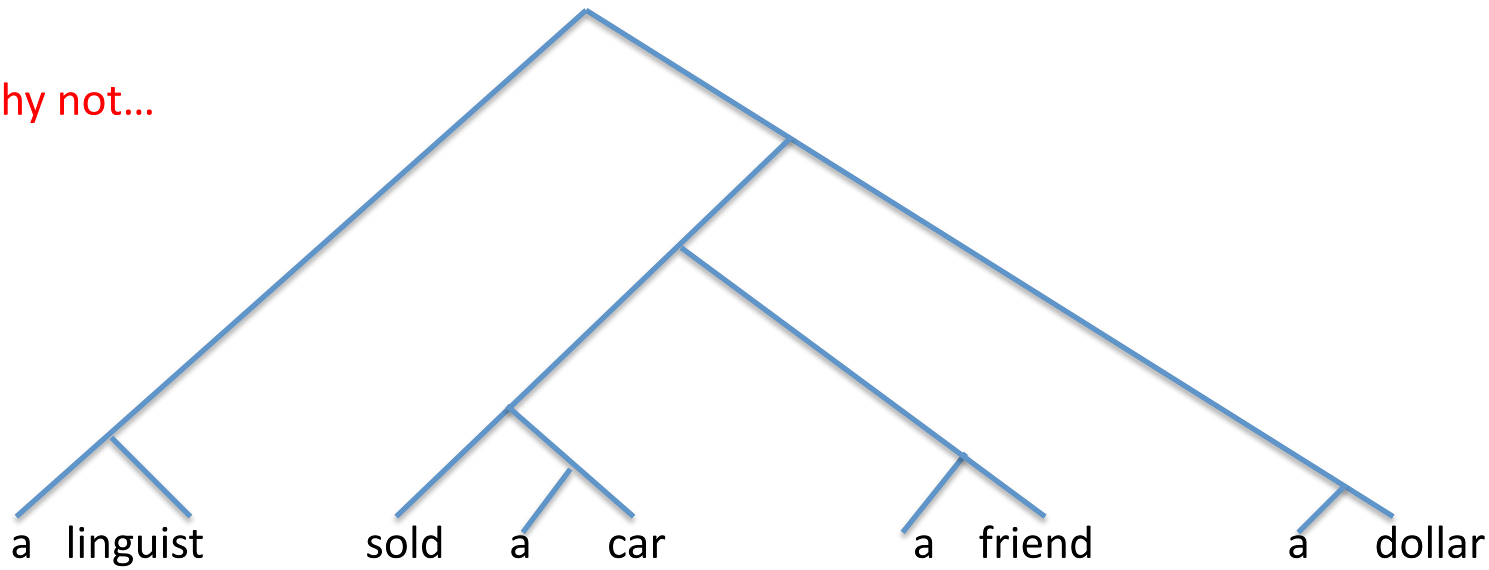


(sold a friend a car for a dollar)

whatever the *order* of arguments,
the concept SOLD, which differs from GAVE,
is plausibly (at least) *tetradic*

Can Human Lexical Items have “Level Four Meanings”?

So why not…



x (she

sold

y this

z him

w that)

↓

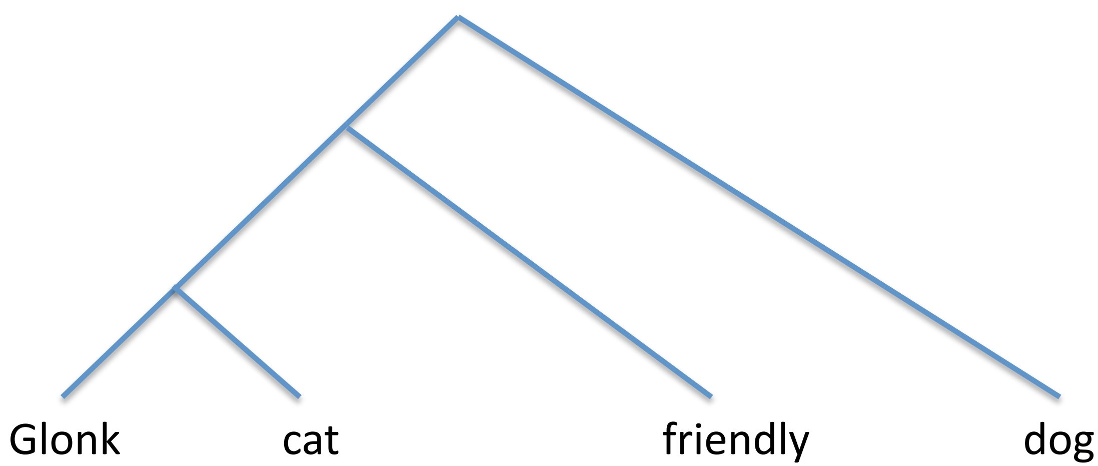
$$\lambda y.\ \lambda z\ .\ \lambda w.\ \lambda x\ .\ x \text{ sold } y \text{ to } z \text{ for } w$$

Can Human Lexical Items have "Level Four Meanings"?

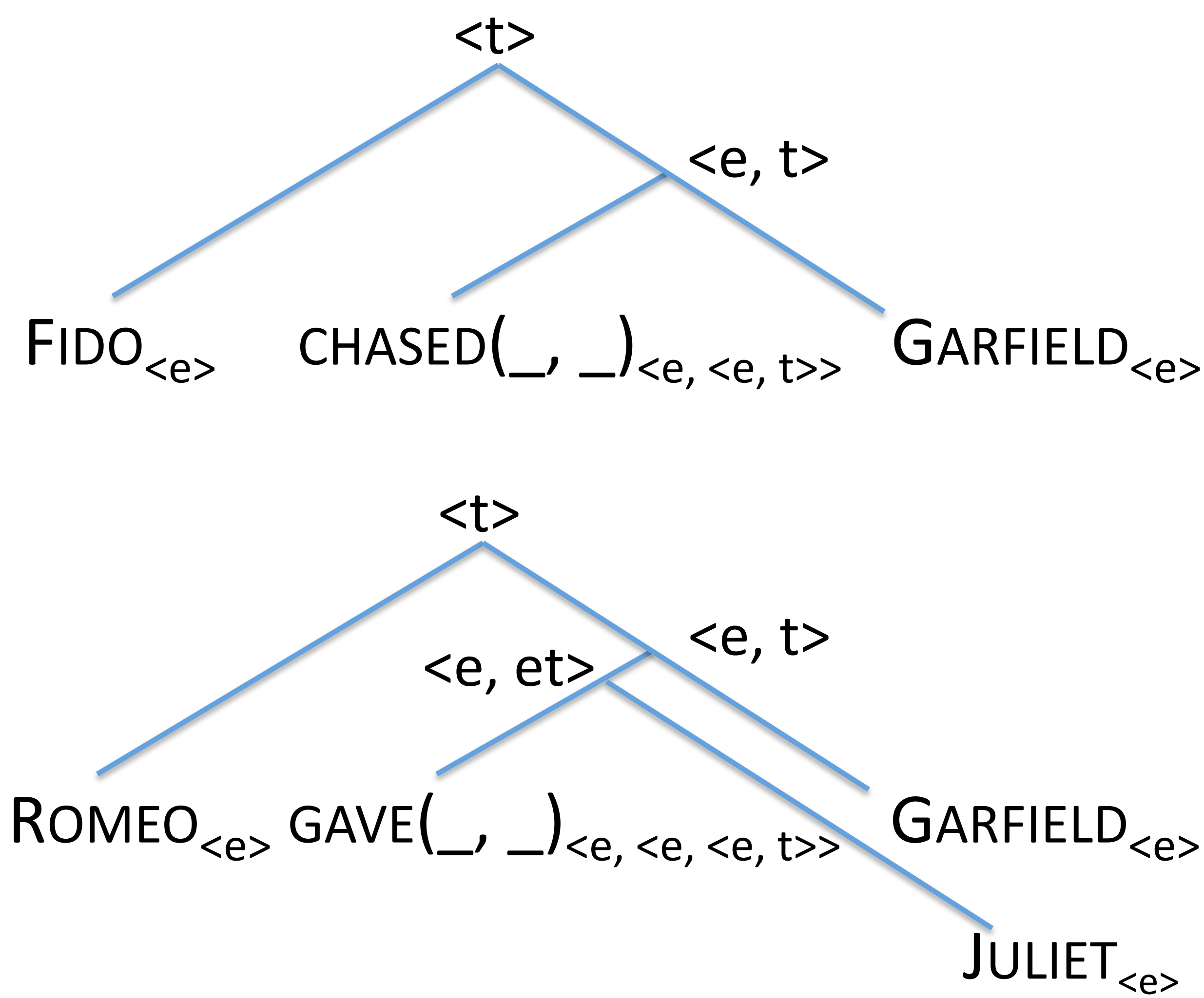
$\lambda Z . \lambda Y. \lambda X . \text{GLONK}(X, Y, Z)$

$\forall x[X(x) \vee Y(x) \vee Z(x)]$

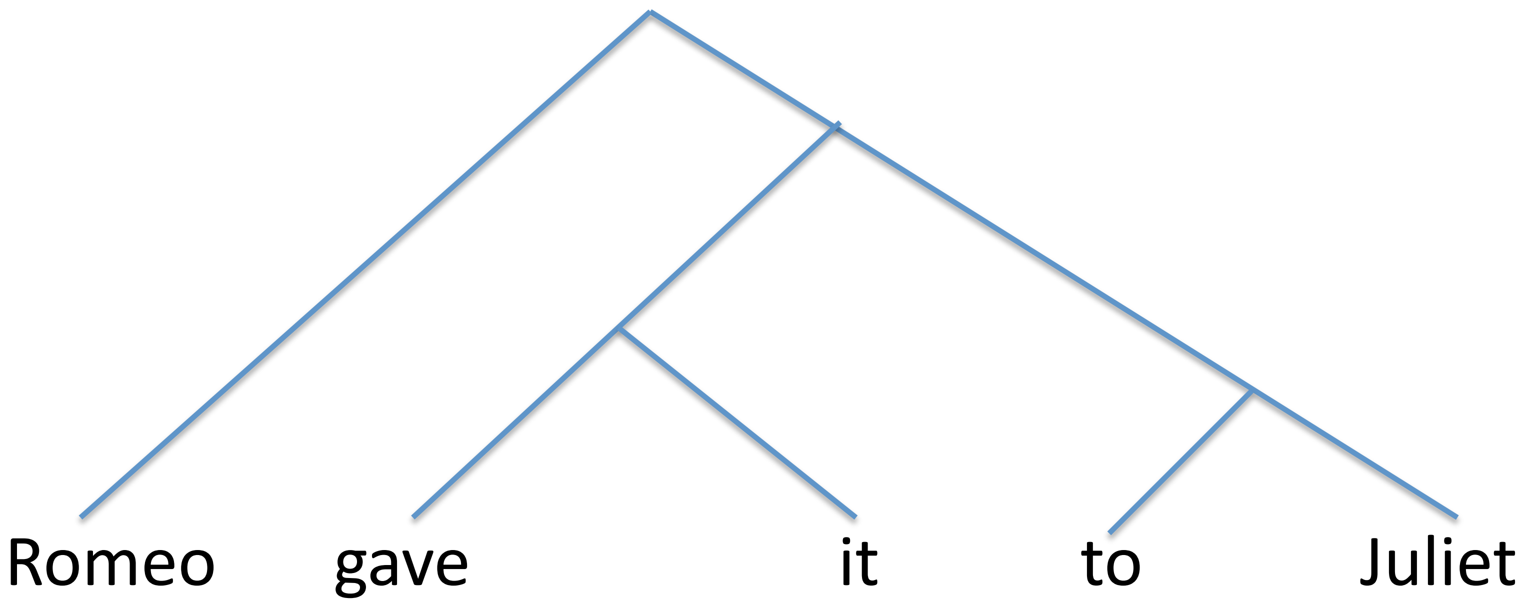
$\exists x[X(x) \,\&\, Y(x)] \,\&\, \exists x[Y(x) \,\&\, Z(x)]$



Can Human Lexical Items have Level Three Meanings?

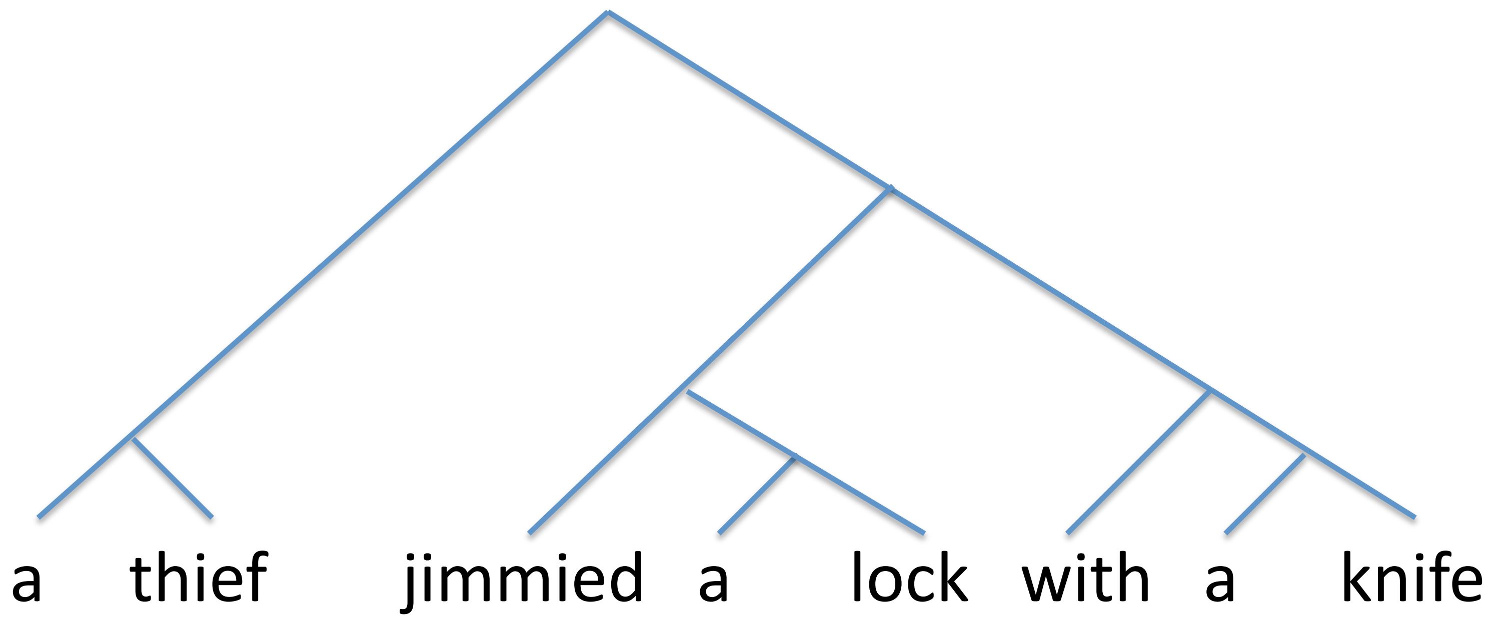


but double-object *constructions* do not show that verbs can have Level Three Meanings

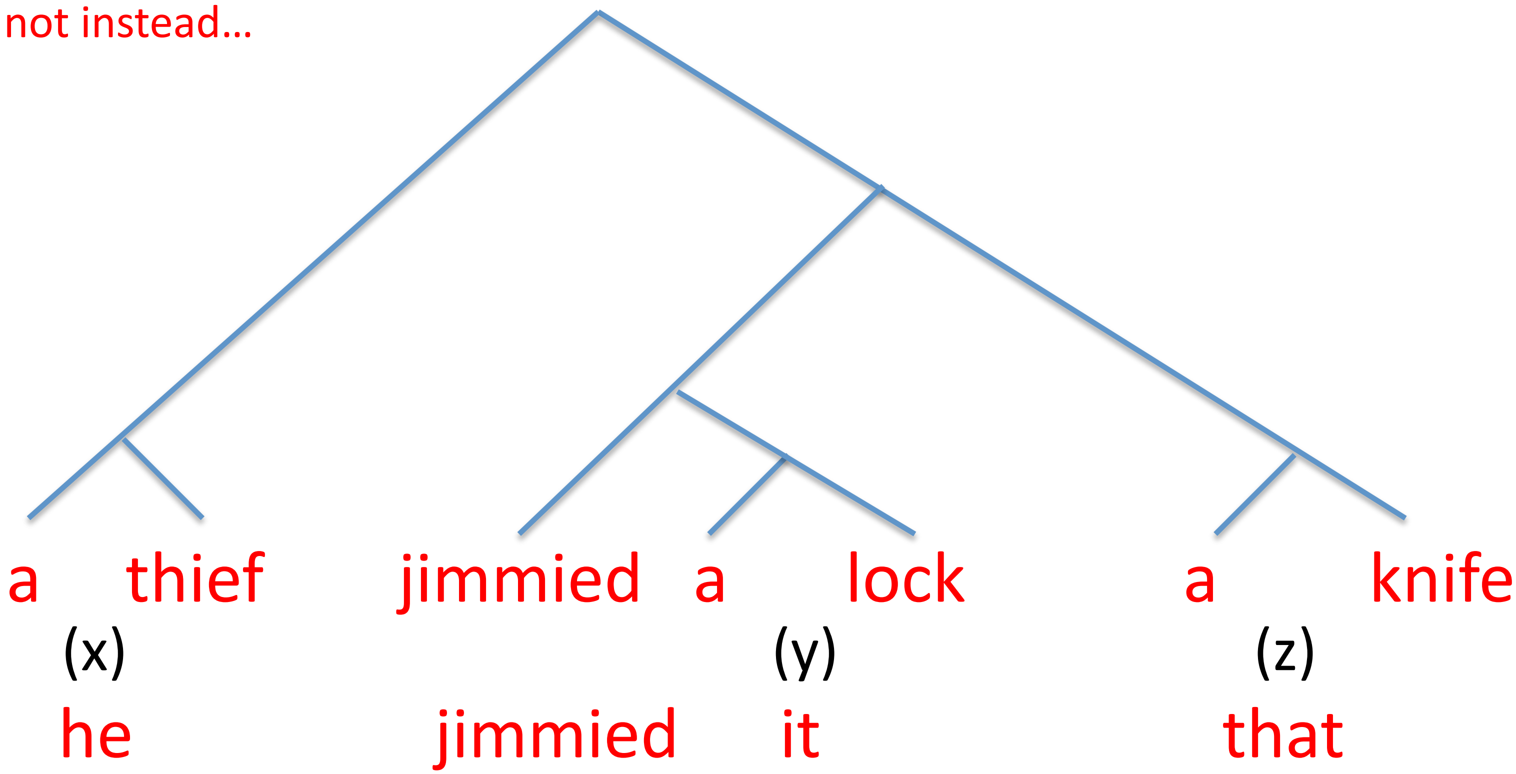


Romeo kicked the rock to Juliet

Romeo kicked Juliet the rock


a
thief
jimmied
a
lock
with
a
knife

Why not instead…

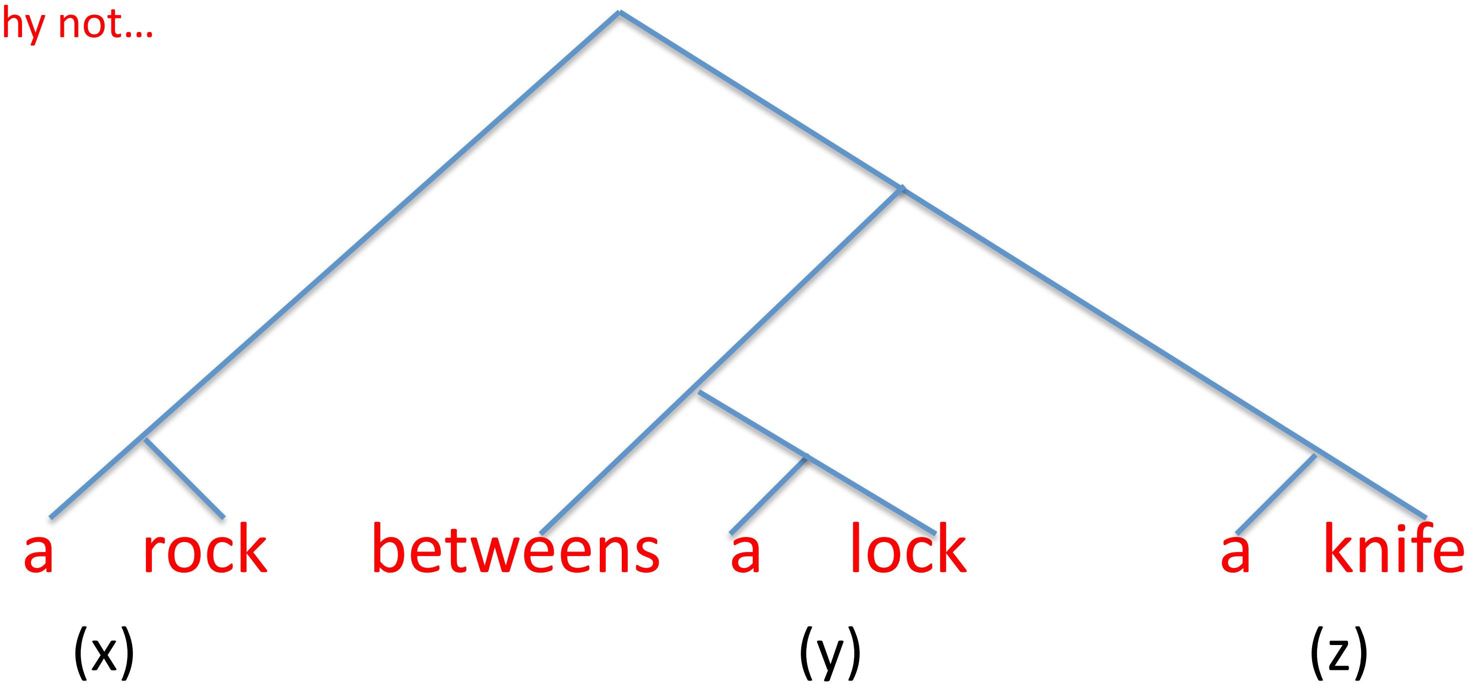


‘jimmied’ ➔ λz. λy . λx . x jimmied y with z

The concept JIMMIED is plausibly (at least) triadic.

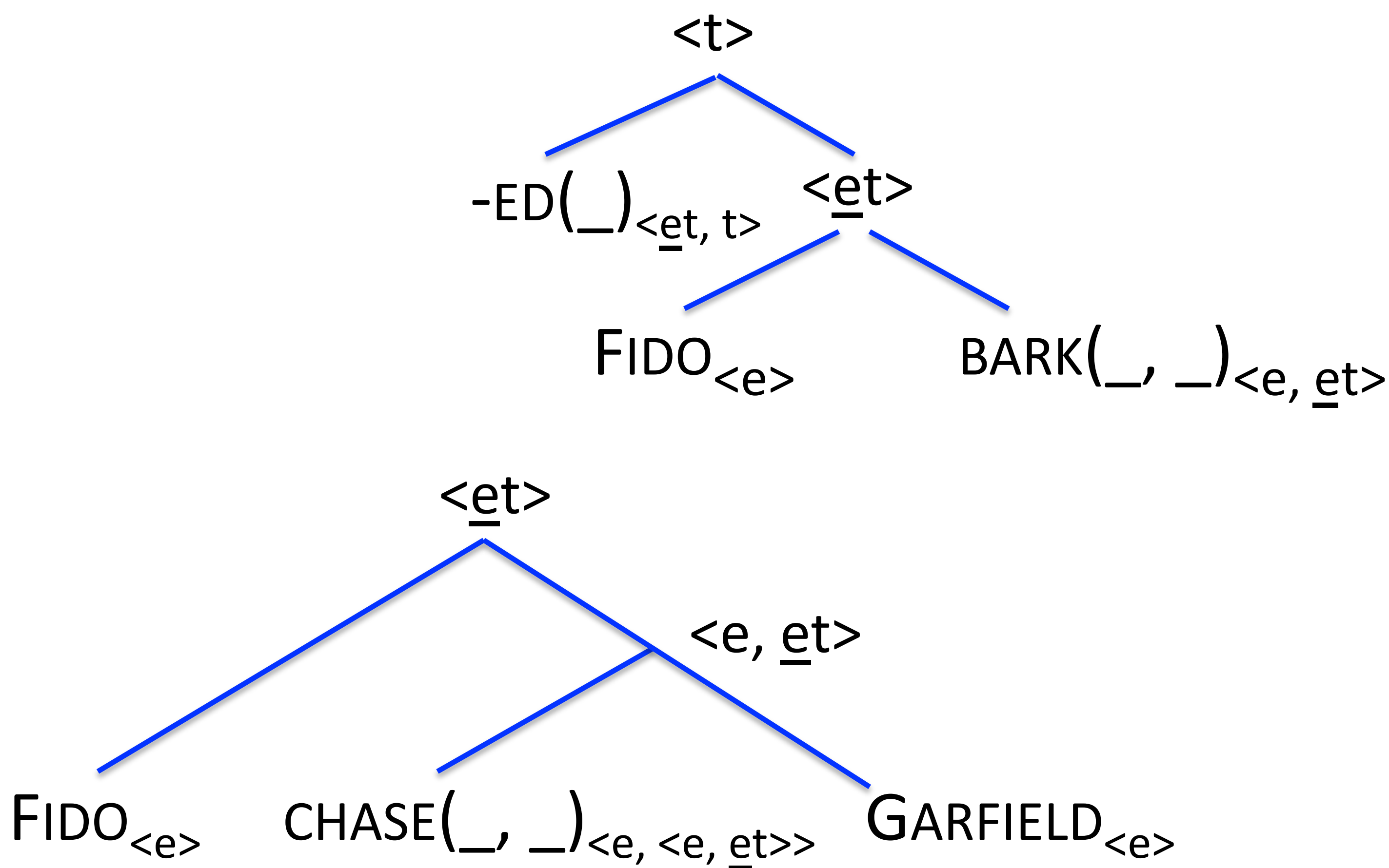
So why *isn’t* the verb of type <e, <e, <et>>>?

Why not…

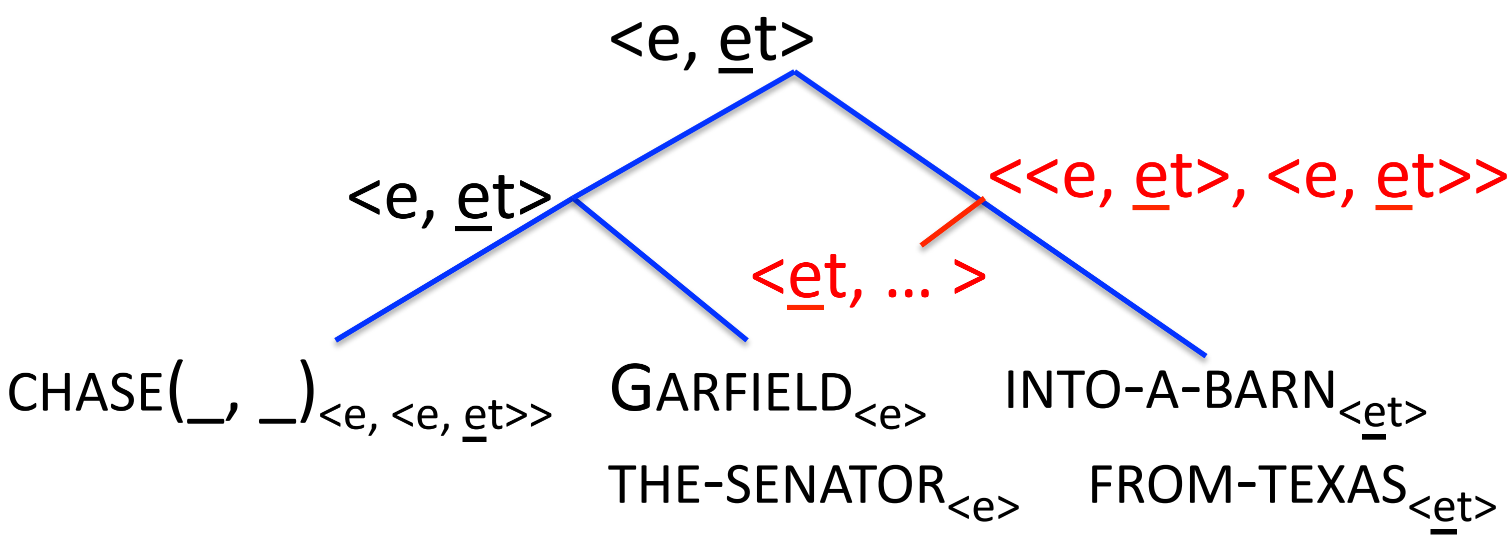


‘betweens’ ➔ λz. λy . λx . x *is* between y *and* z

Still, one might think that many verbs do have Level Three Meanings…

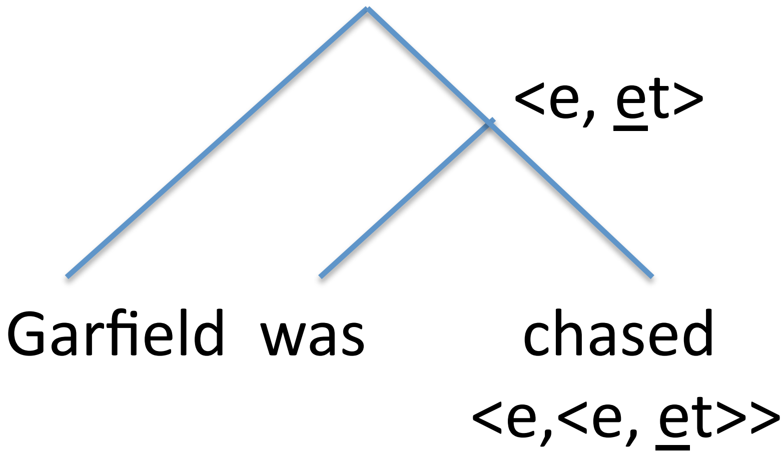


Can Human Lexical Items have Level Three Meanings?

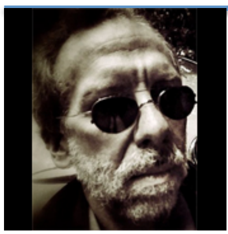


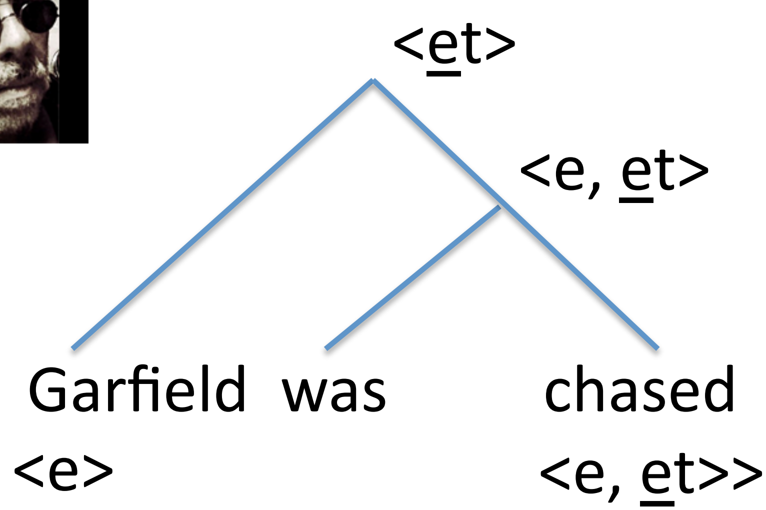
Saying that expressions of type <e, et> can be modified by expressions of type <et> is like positing a covert Level 4 *element*.

And why does the modifier skip over the thing chased, applying instead to the chase?



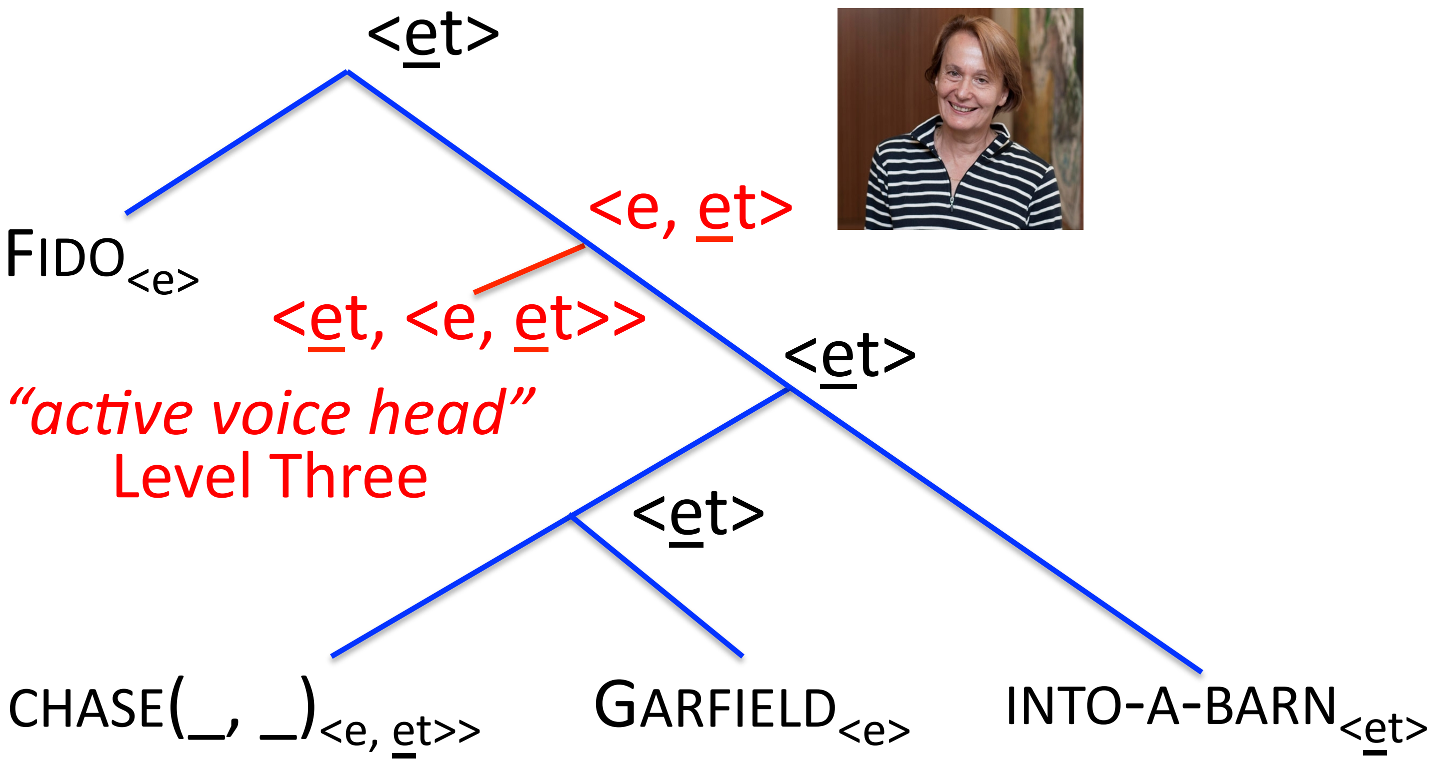
if the meaning of ‘chase’
is at Level Three,
then a “passivizer” would
also be at Level Four:
<<e,<e, et>, <e, et>>



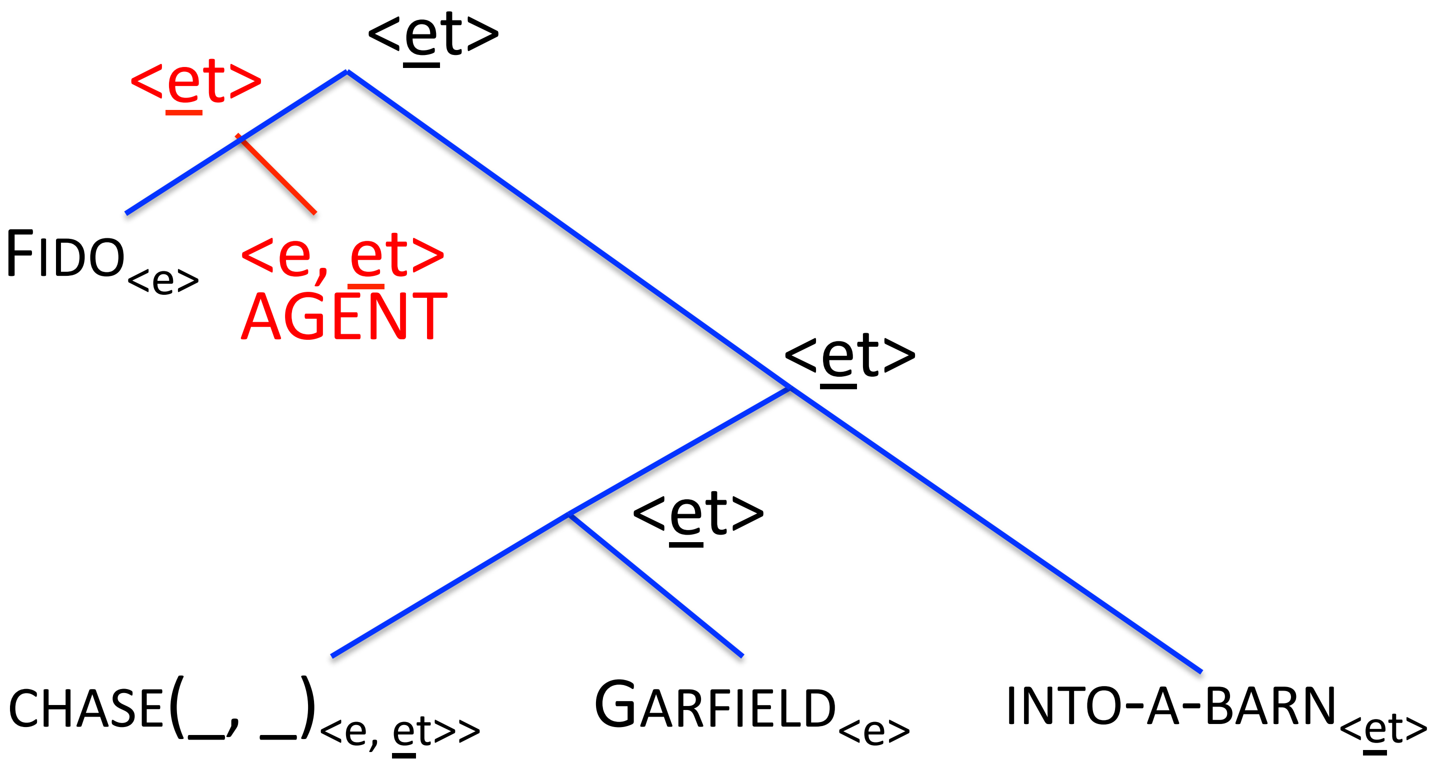


Kratzer and others
“sever” agent-variables
from verb meanings:

‘chase’ ➔
λy. λe . e is a chase of y



But if the posited verb meaning is below Level Three, do we really need the covert Level Three element?


<et>
<et>
FIDO<e>
<e, et>
AGENT
<et>
<et>
CHASE(_, _)<e, et>>
GARFIELD<e>
INTO-A-BARN<et>

What are the Human Meaning Types?

- one familiar answer, via Frege’s conception of *ideal* languages
  - (i) a basic type <e>, for *entity denoters*
  - (ii) a basic type <t>, for *thoughts* or *truth-value denoters*
  - (iii) if <α> and <β> are types, then so is <α, β>

- but is it *independently* plausible that some of our *human* linguistic expressions have meanings of type <e>?
  - -- proper nouns like ‘Tyler’, ‘Burge’, and ‘Pegasus’?
  - -- pronouns like ‘he’, ‘she’, ‘it’, ‘this’, ‘that’ ?

- we know how to Pegasize, and treat names as special cases of monadic predicates

What are the Human Meaning Types?

- one familiar answer, via Frege's conception of ***ideal*** languages

  (i) a basic type <e>, for *entity denoters*

  (ii) a basic type <t>, for *thoughts* or *truth-value denoters*

  (iii) if <α> and <β> are types, then so is <α, β>

- but is it *independently* plausible that some of our *human* linguistic expressions have meanings of type <t>?

  -- which ones? VPs, TPs, CPs?

  -- pronouns like 'he', 'she', 'it', 'this', 'that' ?

- we know (via Tarski) how to treat "sentences" as special cases of monadic predicates

Do Human i-Languages have expressions of type <t>?

S ➔ NP aux VP

```
        T(P)
       /    \
     T       V(P) ➔
   past     /    \
          D(P)    V(P)
          John   /    \
                V      D(P)
               see     Mary
```

Why think *tensed* phrases denote truth values?

V(P) ➔ $\lambda \underline{e}$. $\underline{e}$ is (tenselessly) a John-see-Mary event

Why think the *tense* morpheme

is of type <$\underline{e}$t, t> — $\lambda \underline{E} . \exists \underline{e}[Past(\underline{e})\ \&\ \underline{E}(\underline{e})]$

as opposed to <$\underline{e}$t> or <M> — $\lambda \underline{e} . Past(\underline{e})$

Do Human i-Languages have expressions of type <t>?

```
          T(P)
        /     \
     T         V(P) ➔  λe . e is (tenselessly) a John-see-Mary event
   past       /    \
          D(P)     V(P)
          John    /    \
                 V      D(P)
                see     Mary
```

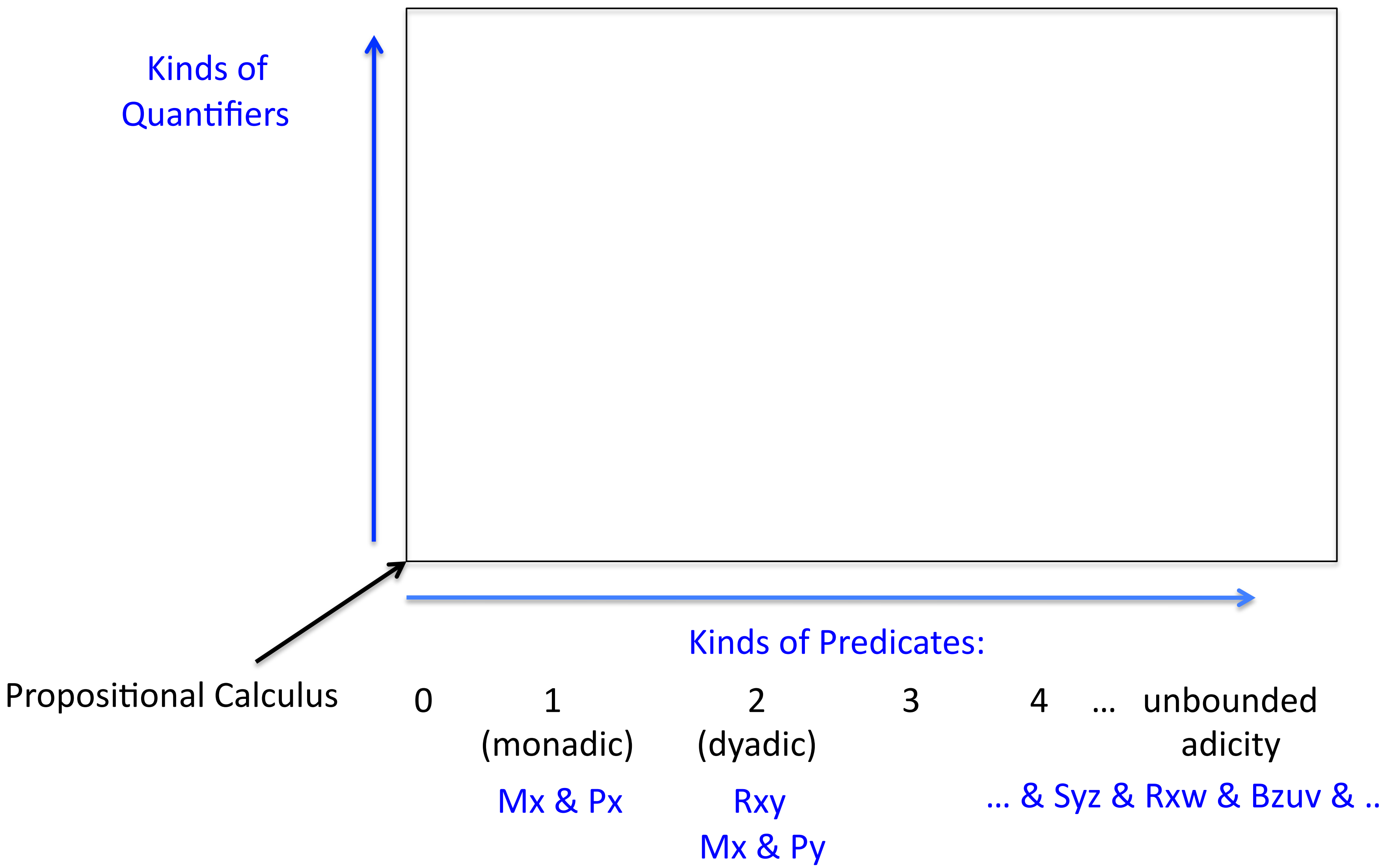
Why think the *tense* morpheme is of type <et, t>

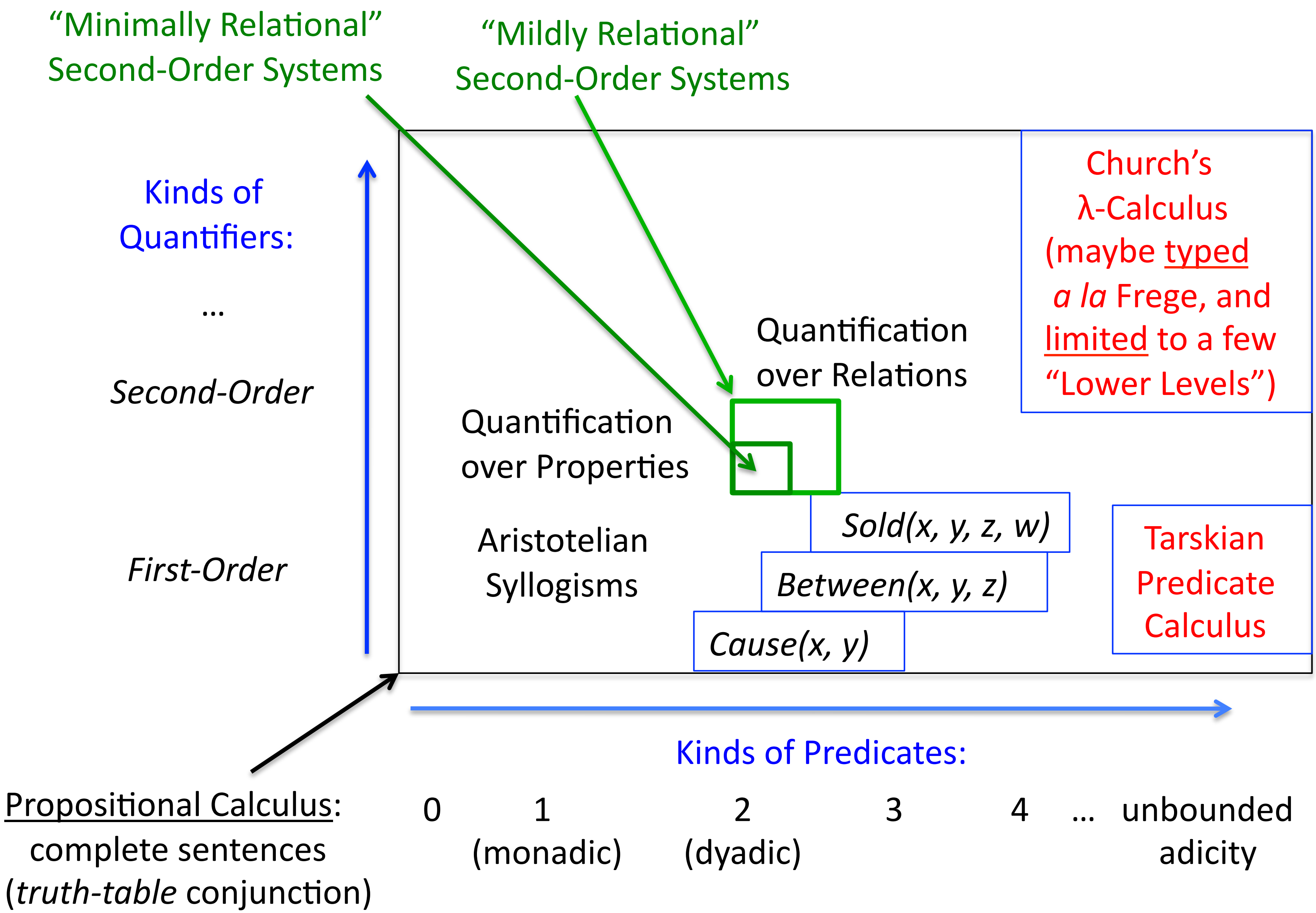
a quantifier...

|

$\lambda\underline{E} . \exists\underline{e}[\text{Past}(\underline{e}) \,\&\, \underline{E}(\underline{e})]$

...that is also a conjunctive adjunct to V?


Kinds of Quantifiers
Propositional Calculus
Kinds of Predicates:
0
1 (monadic)
Mx & Px
2 (dyadic)
Rxy
Mx & Py
3
4
… unbounded adicity
… & Syz & Rxw & Bzuv & …


"Minimally Relational" Second-Order Systems
"Mildly Relational" Second-Order Systems
Kinds of Quantifiers:
...
Second-Order
First-Order
Church's λ-Calculus (maybe typed a la Frege, and limited to a few "Lower Levels")
Quantification over Relations
Quantification over Properties
Aristotelian Syllogisms
Sold(x, y, z, w)
Between(x, y, z)
Cause(x, y)
Tarskian Predicate Calculus
Kinds of Predicates:
Propositional Calculus: complete sentences (truth-table conjunction)
0
1 (monadic)
2 (dyadic)
3
4
... unbounded adicity

Plan for Rest of the Talk

- Characterize a notion of "Minimally Relational"
- Describe a Possible Language that is Minimally Relational and (correlatively) "Minimally Interesting" in this respect
- Suggest that while Human Meanings may be *a little* more interesting, they approximate Minimal Relationality
- End with reminders of some other respects in which Human Languages seem to be Minimally Interesting, and suggest that semantic typology is yet another case

Flavors of Recursion

- Some recursive procedures are very, very, ... , very boring
- Others generate more interesting

  [phrases [within [phrases [within [phrases …]]]]]

- And some allow for displacement of a sort

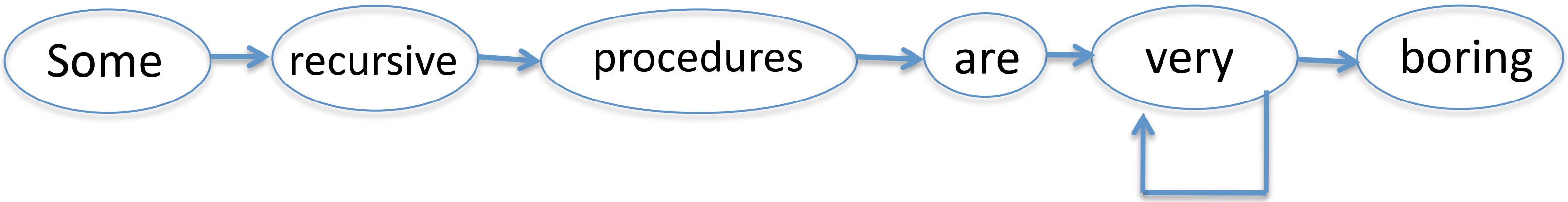
  that permits construction of relative clauses

  like ‘who saw Juliet’ and ‘who Romeo saw’,

  whose elements can be systematically recombined

  to form boundlessly many expressions

  that allow for displacement…



N → phrases

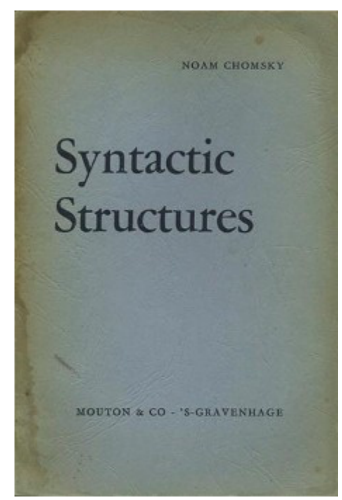
NP → N

P → within

PP → P NP   PP → within NP → within N → within phrases

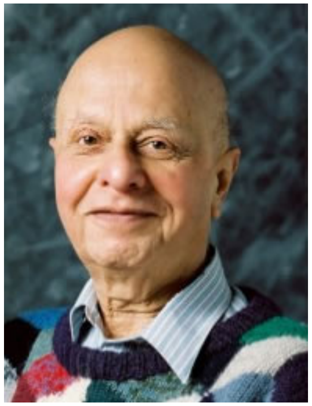
NP → N PP   NP → N within phrases → phrases within phrases

S → NP aux VP → Romeo did see Juliet →
Romeo saw Juliet → Romeo saw *who* →
who Romeo saw *t* ← CP

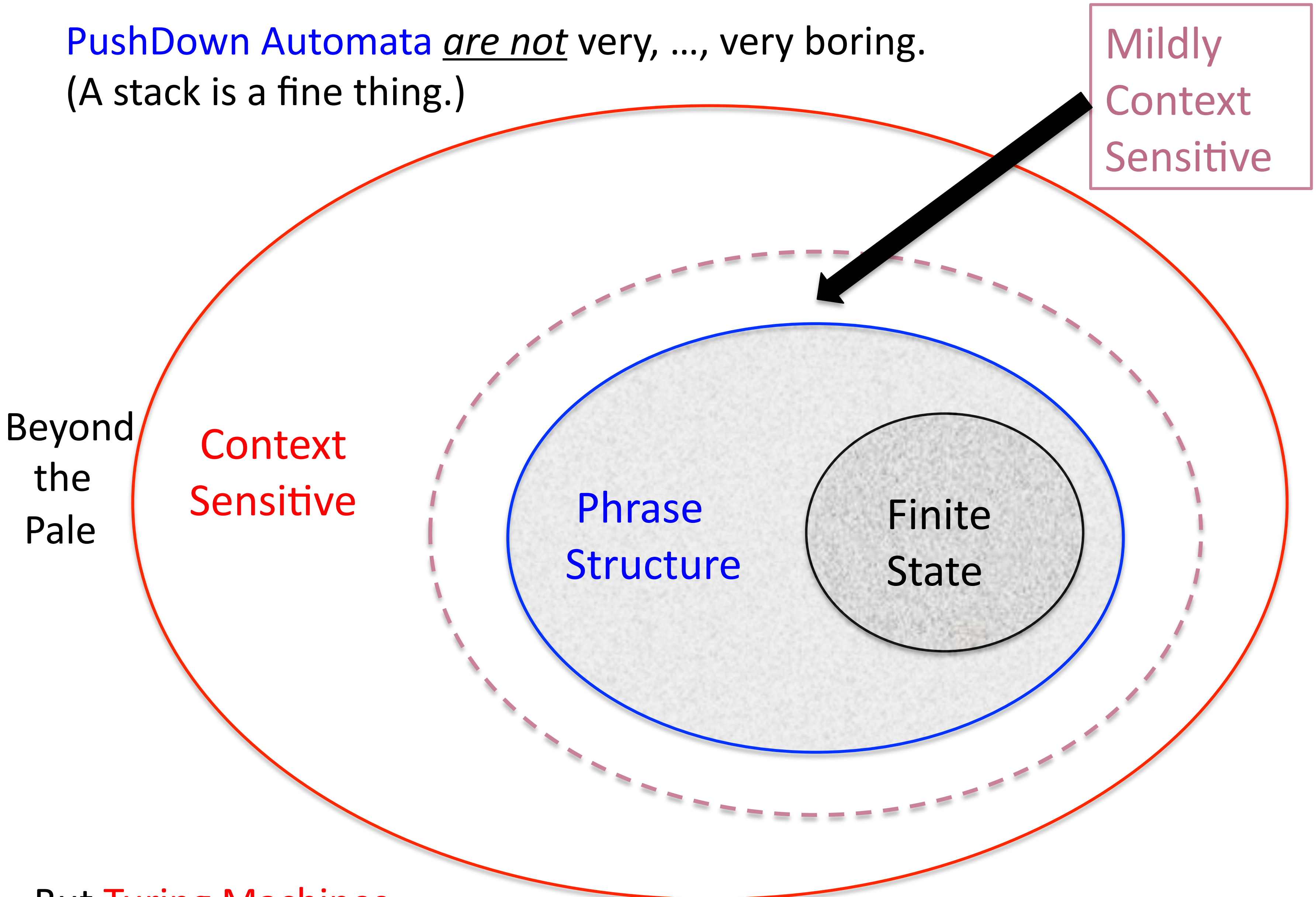


Ways of Generating Lots of Expressions

- Finite State (Markovian)
- Phrase Structure (“Context Free”)
- Transformational
  - but humanly constrained (“mildly” context sensitive)
  - not so constrained (“pret-ty” context sensitive)
  - computable but otherwise unconstrained

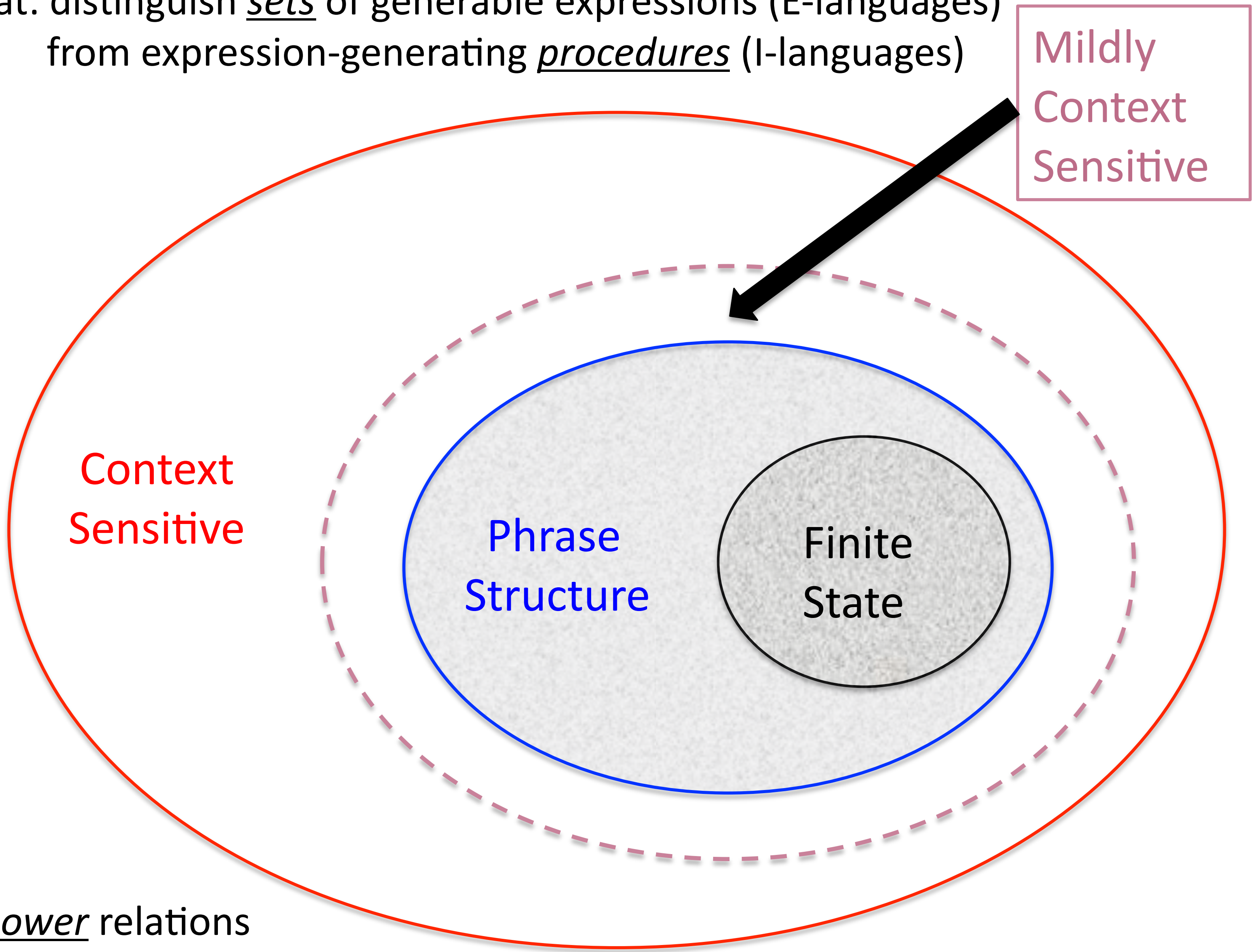


PushDown Automata *<u>are not</u>* very, …, very boring.
(A stack is a fine thing.)



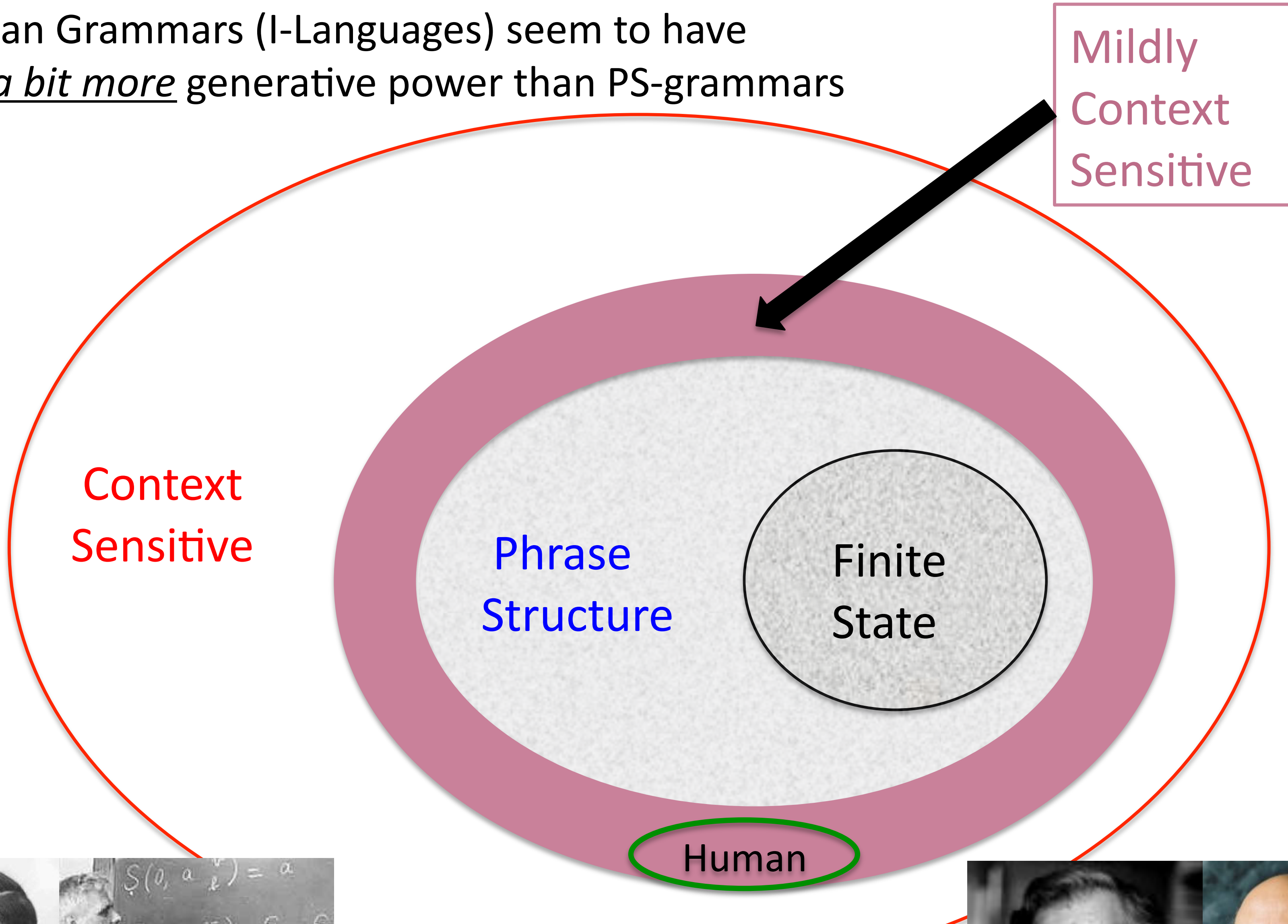
But Turing Machines
(with limited tape) can do *<u>a lot</u>* more.

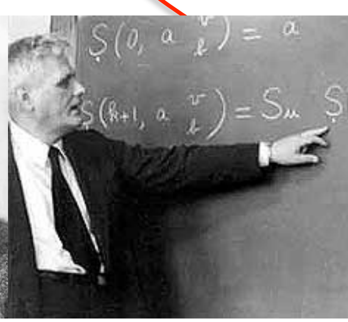
Caveat: distinguish *sets* of generable expressions (E-languages)
from expression-generating *procedures* (I-languages)



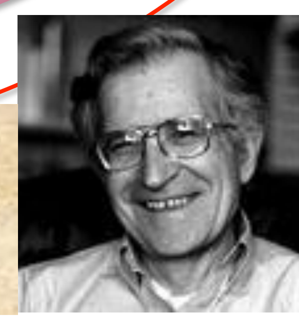
the *power* relations
reflect the available operations: with regard to *generative capacity*,
CS-grammars > PS-grammars > FS-grammars

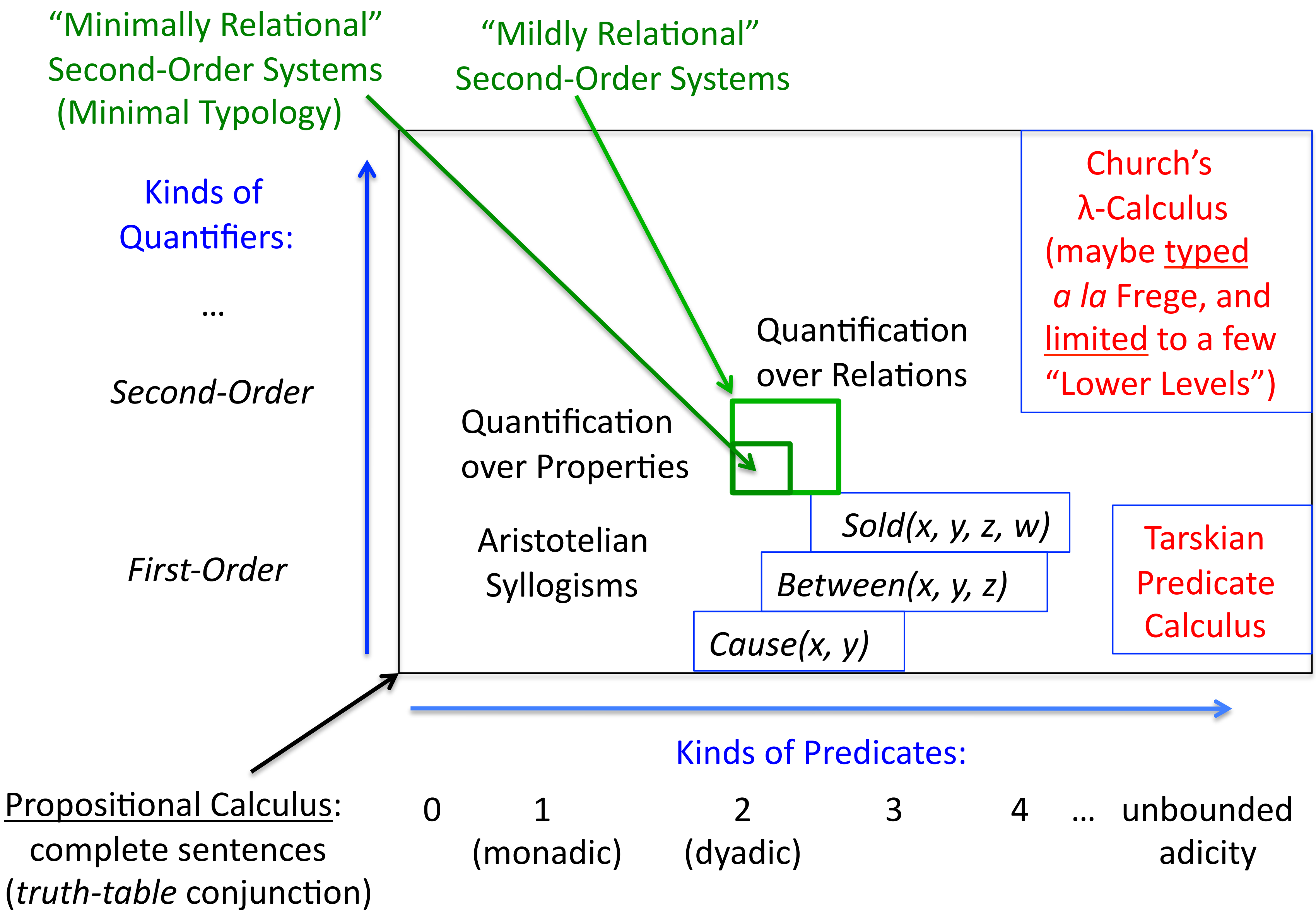
Human Grammars (I-Languages) seem to have
a bit more generative power than PS-grammars



This locates Human Languages in a "Computational Space." Can they be located in a "Semantic Space"?



“Minimally Relational”
Second-Order Systems
(Minimal Typology)
“Mildly Relational”
Second-Order Systems
Kinds of
Quantifiers:
...
Second-Order
First-Order
Church’s
λ-Calculus
(maybe typed
a la Frege, and
limited to a few
“Lower Levels”)
Quantification
over Relations
Quantification
over Properties
Aristotelian
Syllogisms
Sold(x, y, z, w)
Between(x, y, z)
Cause(x, y)
Tarskian
Predicate
Calculus
Kinds of Predicates:
Propositional Calculus:
complete sentences
(truth-table conjunction)
0
1
(monadic)
2
(dyadic)
3
4
...
unbounded
adicity

a basic type <e>, for entity denoters
a basic type <t>, for truth-value denoters
if <α> and <β> are types, then so is <α, β>

at Level 5,
more than 5 x 10^{12}

0.	<e>	<t>	(2) types at Level Zero
1.	<e, e> <e, t>	<t, e> <t, t>	(4) at Level One, all <0, 0>
2.	eight of <0, 1> sixteen of <1, 1>	eight of <1, 0>	(32), including <e, et> and <et, t>
3.	64 of <0, 2> 128 of <1, 2> 1024 of <2, 2>	64 of <2, 0> 128 of <2, 1>	(1408), including <e, <e, et>>; <et, <et, t>>; and <<e, et>, t>
4.	2816 of <0, 3> 5632 of <1, 3> 45,056 of <2, 3> 1,982,464 of <3, 3>	2816 of <3, 0> 5632 of <1, 3> 45,056 of <3, 2>	(2,089,472), including <e, <e, <e, <et>> and <<e, et>, <<e, et>, t>

Thanks,

and thanks to Jim