

A Computational Implementation of Syntactic Binding Theory in the Minimalist Program

Sandiway Fong
Departments of Linguistics and Computer Science
University of Arizona
Tucson AZ USA

Joint work with *Jason Ginsburg*
(University of Aizu, Japan)

Paper and Derivations Available

from my homepage

- *(just google my first name)*
- <http://dingo.sbs.arizona.edu/~sandiway/mpp/>

Computation with doubling constituents: Pronouns and antecedents in Phase Theory

Sandiway Fong Jason Ginsburg
University of Arizona University of Aizu

Abstract

In this paper, we develop a computational implementation of (syntactic) Binding Theory compatible with basic assumptions from the Minimalist Program. We take as our starting point the basic Merge/Move operations plus the probe-goal Case agreement system of Chomsky (2000, 2001) together with Kayne's (2002) proposal that pronoun-antecedent coreference relations originate with a base-generated pronominal/r-expression doubling constituent. We propose that an anaphor and r-expression antecedent originate in a DP doubling constituent that is a (strong) phase, with the phase head 'self', and a pronoun and r-expression antecedent originate in a DP doubling constituent that is not a phase. Under standard assumptions about locality and Phase Theory, the r-expression half of the doubling constituent must separate from the pronominal component in a timely fashion and undergo movement to be licensed independently (i.e. receive its own theta role and have its case feature valued). We propose that movement of the r-expression is constrained by an operation of Last Resort, whereby the r-expression, under appropriate circumstances, can be remerged into a derivation and licensed. We demonstrate, via a computational implementation, how our proposals derive the classic asymmetry in distribution between pronouns and anaphors for mono- and bi-clausal sentences, ECM, picture DPs and other constructions.

1 Introduction

As part of the biolinguistics perspective, the Minimalist Program (MP) (Chomsky 1995) focuses on simple and optimal solutions to the problem of the nature of human language. It is expected that considerations of efficient computation (within the constraints of the biological system) should contribute to and help explain the shape or restrictions on the space of possible human languages (cf. Chomsky 2005). An uncontroversial property of this computational system is that it should be recursive, i.e. in principle allow the unbounded combination of a limited number of lexical primitives to form an infinite variety of specific structures that encode the variety of language phenomena. Within the strictures of the MP, it is proposed that simple recursive merge (external and internal, i.e. displacement) suffices to generate an arbitrary number of structures, and the agree relation that obtains between probes and goals serves to restrict the possible instances (Chomsky 2001). Given that recursive merge can generate arbitrarily complex structures, efficient computation dictates that

Example:

(3) John[1] thinks he[1] is smart

Derivation steps:

Introduction: Data

- Classic Binding Theory data:
 - *John_i praises him_i
 - John_i praises himself_i
 - John_i thinks he_i is smart
 - *He_i thinks John_i is smart
 - *John_i thinks himself_i is smart
 - *John_i thinks that Mary likes himself_i
 - John_i considers himself_i to be intelligent
 - *John_i considers him_i to be intelligent

Introduction: Data

- Classic Binding Theory data:

John_i likes his_i dog

*John_i likes himself_i's dog

?*Hannah_i found a picture of her_i

Hannah_i found a picture of herself_i

?*Hannah found Peter_i's picture of him_i

Hannah found Peter_i's picture of himself_i

Hannah_i found Peter's picture of her_i

Hannah_i found Peter's picture of herself_i

Note: *picture*-DP judgments from Keller & Asudeh's (2001) study

**Binding
Conditions:
A, B and C**

Introduction: Theory

Assumptions:

- Minimalist Program Chomsky (2000, 2001)
 - merge (external/internal)
 - uninterpretable/interpretable features
 - probe-goal search
 - (strong) phase boundaries (v^* , c)

add: d headed by *self*

possessive 's

“sufficiently complex”

Introduction: Theory

treeserver.tcl

TREE VIEWER

Zoom: (x1) (x4)

1. theta merge V & N
2. merge v & V
3. theta merge N & v
4. merge T & v
5. move to spec-T
6. merge C & T

```
graph TD; V1[V] --- V2[V like]; V1 --- n[n mary];
```

*Object Mary is first merged with V like
its theta role is identified here*

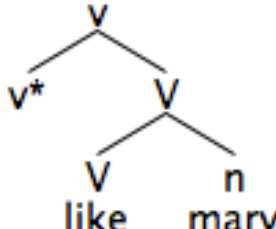
Introduction: Theory

treeviewer.tcl

TREE VIEWER

Zoom: (x1) (x4)

1. theta merge V & N
2. merge v & V
3. theta merge N & v
4. merge T & v
5. move to spec-T
6. merge C & T



Probe [v*] agrees with goal [n mary]

ϕ -features of v are valued by Mary
Mary gets accusative Case from v**

Introduction: Theory

treeviewer.tcl

TREE VIEWER

Zoom: (x1) (x4)

1. theta merge V & N
2. merge v & V
3. theta merge N & v
4. merge T & v
5. move to spec-T
6. merge C & T

John is first merged into spec-v
the VP-internal subject position
its theta role is identified here*

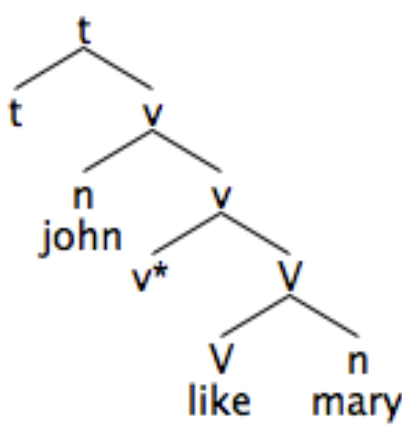
Introduction: Theory

treeviewer.tcl

TREE VIEWER

Zoom: (x1) (x4)

1. theta merge V & N
2. merge v & V
3. theta merge N & v
4. merge T & v
5. move to spec-T
6. merge C & T



Probe [t] agrees with goal [n john]

*ϕ -features of t are valued by John
John gets nominative Case from t*

Introduction: Theory

treeviewer.tcl

TREE VIEWER

Zoom: (x1) (x4)

1. theta merge V & N
2. merge v & V
3. theta merge N & v
4. merge T & v
5. move to spec-T
6. merge C & T

copy theory: only the first occurrence of John is pronounced

Introduction: Theory

treeviewer.tcl

TREE VIEWER

Zoom: (x1) (x4)

1. theta merge V & N
2. merge v & V
3. theta merge N & v
4. merge T & v
5. move to spec-T
6. merge C & T

```
graph TD
    c1[c] --- c2[c]
    c1 --- t1[t]
    t1 --- n1[n]
    n1 --- john1[john]
    t1 --- t2[t]
    t2 --- v1[v]
    v1 --- n2[n]
    n2 --- john2[john]
    v1 --- v2[v]
    v2 --- v3[v*]
    v2 --- V[V]
    V --- V2[V]
    V2 --- like[like]
    V --- n3[n]
    n3 --- mary[mary]
```

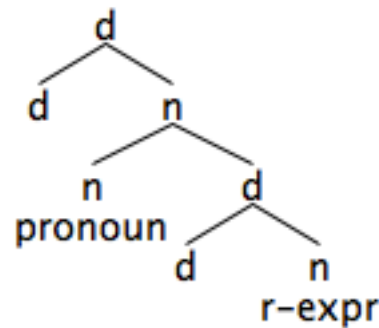
Introduction: Theory

- Add Kayne's (2002) proposal
 - pronominal and r-expression
 - doubling constituent (DC): [spec head], e.g. [John he]
 - Stipulation: spec can move out of DC after DC moves first
 [_{TP} John thinks [_{TP} [John he] is [_{AP} smart [John he]]]]

Other related work:
 Zwart (2002)
 Heinat (2003)

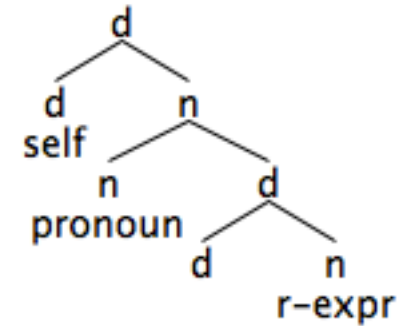
- Our implementation of the DC:

1. merge D & N
2. merge N & D
3. merge D & N



Both pronoun and r-expr

1. uninterpretable Case
2. interpretable φ -features
3. need theta-role



reflexive anaphor
 4. phase boundary

John thinks he is smart

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge C & T
9. merge V & C
10. merge v* & V
11. LR move to v*
12. merge T & v
13. move to spec-T
14. merge C & T

```
graph TD
    d1[d] --- d2[d]
    d1 --- n1[n]
    d2 --- n2[n]
    d2 --- he[he]
    n1 --- d3[d]
    n1 --- n3[n]
    d3 --- d4[d]
    d3 --- n4[n]
    n4 --- john[john]
```

Scoreboard	<i>he</i>	<i>John</i>
needs	theta, Case	theta, Case
licensed		

John thinks he is smart

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge C & T
9. merge V & C
10. merge v* & V
11. LR move to v*
12. merge T & v
13. move to spec-T
14. merge C & T

```
graph TD
    a1[a] --- a2[a]
    a1 --- d1[d]
    a2 --- smart[smart]
    d1 --- d2[d]
    d1 --- n1[n]
    d2 --- he[he]
    n1 --- d3[d]
    n1 --- n2[n]
    d3 --- john1[John]
    n2 --- john2[John]
```

Scoreboard	<i>he</i>	<i>John</i>
needs	Case	theta, Case
licensed	theta	

John thinks he is smart

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge C & T
9. merge V & C
10. merge v* & V
11. LR move to v*
12. merge T & v
13. move to spec-T
14. merge C & T

The syntax tree for the sentence "John thinks he is smart" is shown. The root node is *v*, which branches into *v* (labeled "be") and *a*. The *a* node branches into *a* (labeled "smart") and *d*. The *d* node branches into *d* and *n*. The *n* node branches into *n* (labeled "he") and *d*. The *d* node branches into *d* and *n* (labeled "john").

Scoreboard	<i>he</i>	<i>John</i>
needs	Case	theta, Case
licensed	theta	

John thinks he is smart

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge C & T
9. merge V & C
10. merge v* & V
11. LR move to v*
12. merge T & v
13. move to spec-T
14. merge C & T

Probe [t] agrees with goal [d[d][n[n he][d!case!arg[d][n john]]]]

Scoreboard	<i>he</i>	<i>John</i>
needs		theta, Case
licensed	theta, Case	

John thinks he is smart

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge C & T
9. merge V & C
10. merge v* & V
11. LR move to v*
12. merge T & v
13. move to spec-T
14. merge C & T

Scoreboard	<i>he</i>	<i>John</i>
needs		theta, Case
licensed	theta, Case	

John thinks he is smart

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge C & T
9. merge V & C
10. merge v* & V
11. LR move to v*
12. merge T & v
13. move to spec-T
14. merge C & T

Scoreboard	<i>he</i>	<i>John</i>
needs		theta, Case
licensed	theta, Case	

John thinks he is smart

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge C & T
9. merge V & C
10. merge v* & V
11. LR move to v*
12. merge T & v
13. move to spec-T
14. merge C & T

phase boundary

Scoreboard	<i>he</i>	<i>John</i>
needs		theta, Case
licensed	theta, Case	

John thinks he is smart

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge C & T
9. merge V & C
10. merge v* & V
11. LR move to v*
12. merge T & v
13. move to spec-T
14. merge C & T

**Last Resort (LR):
theta merge**

Scoreboard	<i>he</i>	<i>John</i>
needs		Case
licensed	theta, Case	theta

John thinks he is smart

treeviewer.tcl

TREE VIEWER (January 2010)

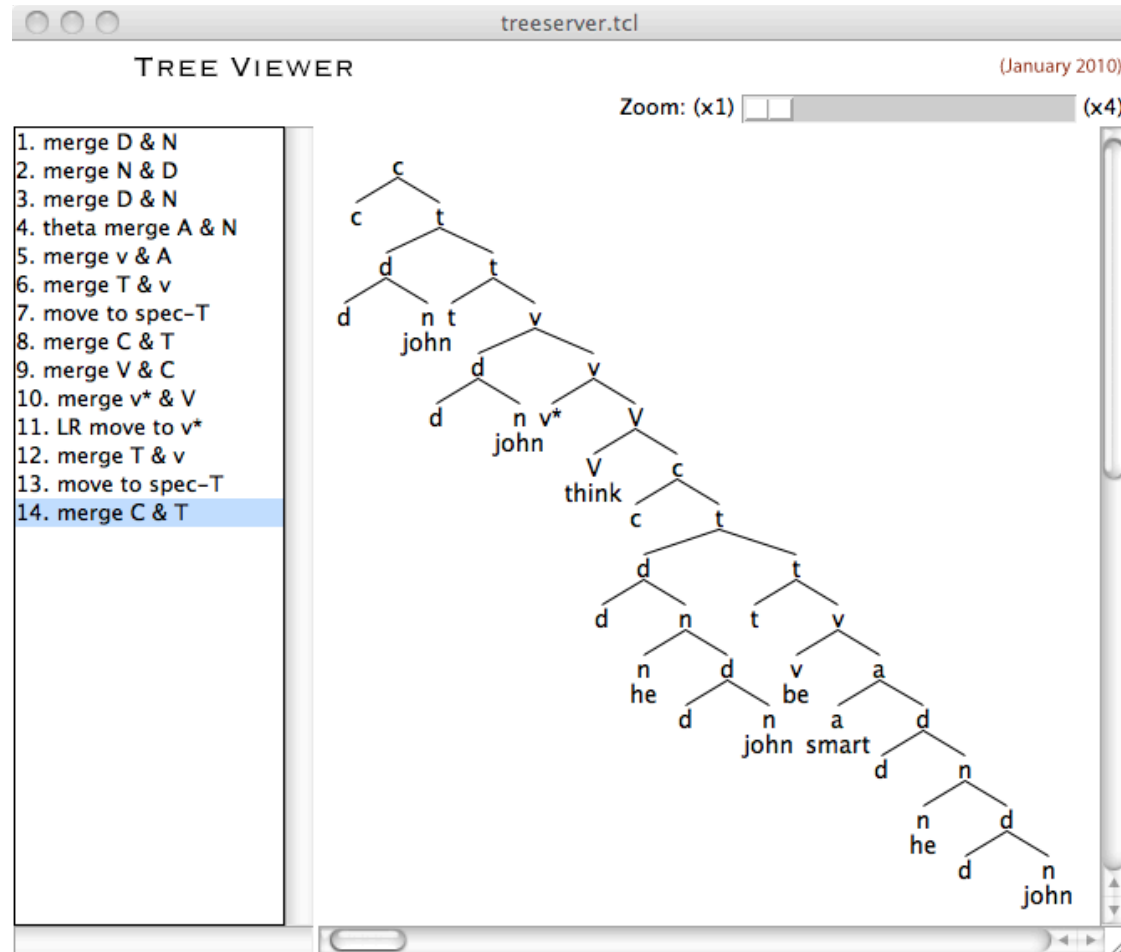
Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge C & T
9. merge V & C
10. merge v* & V
11. LR move to v*
12. merge T & v
13. move to spec-T
14. merge C & T

Scoreboard	<i>he</i>	<i>John</i>
needs		
licensed	theta, Case	theta, Case

Probe [t] agrees with goal [d[d][n john]]

John thinks he is smart



*John praises him

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge V & N
5. merge v* & V

```
graph TD
    d1[d] --- d2[d]
    d1 --- n1[n]
    d2 --- n2[n]
    d2 --- he[he]
    n1 --- d3[d]
    n1 --- n3[n]
    d3 --- d4[d]
    d3 --- john[john]
    n3 --- n4[n]
    n3 --- praises[praises]
```

Scoreboard	<i>he</i>	<i>John</i>
needs	theta, Case	theta, Case
licensed		

*John praises him

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge V & N
5. merge v* & V

```
graph TD
    V1[V] --- V2[V]
    V1 --- d1[d]
    V2 --- praise[praise]
    V2 --- d2[d]
    d1 --- n1[n]
    n1 --- he[he]
    n1 --- d3[d]
    d3 --- d4[d]
    d3 --- n2[n]
    n2 --- john[john]
```

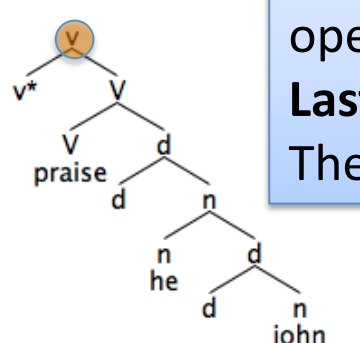
Scoreboard	<i>he</i>	<i>John</i>
needs	Case	theta, Case
licensed	theta	

*John praises him

treeserv

TREE VIEWER

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge V & N
5. merge v* & V



Probe [v*] agrees with goal [d[d][n[n he][d!case!arg[d][n john]]]]

Now suppose only way r-expr *John* can extract out of DC is via that operation ...

Last Resort (LR): theta merge

Then derivation stops (*crashes*)

Scoreboard	<i>he</i>	<i>John</i>
needs		theta, Case
licensed	theta, Case	

John praises himself

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge V & N
5. merge v* & V
6. LR move to v*
7. merge T & v
8. move to spec-T
9. merge C & T

```
graph TD; v_star((v*)) --- V((V)); v_star --- v_star_spec((v*)); V --- praise((praise)); V --- d((d)); d --- self((self)); d --- n1((n)); n1 --- he((he)); n1 --- d2((d)); d2 --- john((john));
```

add: phase boundary
(d when headed by *self*)

Probe [v*] agrees with goal [d[d self][n[n he][d!case!arg[d][n john]]]]

John praises himself

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) [] (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge V & N
5. merge v* & V
6. LR move to v*
7. merge T & v
8. move to spec-T
9. merge C & T

Last Resort (LR):
theta merge

The diagram shows a syntax tree for the sentence "John praises himself". The root node is **v**. Its left child is **d** (circled in orange), which branches into **d** and **n**. The **n** child of this **d** is **john**. The right child of the root **v** is **v**. This **v** branches into **V** and **v***. The **v*** child is **john**. The **V** child branches into **V** and **d** (circled in orange). The **V** child of this **V** is **praise**. The **d** child branches into **d** and **n**. The **d** child of this **d** is **self**. The **n** child branches into **n** and **d**. The **n** child of this **n** is **he**. The **d** child branches into **d** and **n**. The **d** child of this **d** is **john**. The **n** child of this **d** is **john**.

Introduction: Data

- Classic Binding Theory data:
 - *John_i praises him_i
 - John_i praises himself_i
 - John_i thinks he_i is smart
 - *He_i thinks John_i is smart
- Kayne's (2002) proposal
 - DC: [spec head], e.g. [*John he*]
 - only the specifier may move (not the head)

*He_i thinks John_i is smart

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) [] (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge C & T
9. merge V & C
10. merge v* & V
11. LR move to v*
12. merge T & v
13. move to spec-T
14. merge C & T

phase boundary

he is the head in
[_d d [_n he [_d d [_n John]]]]

Scoreboard	he	John
needs		theta, Case
licensed	theta, Case	

Introduction: Data

- Classic Binding Theory data:
 - *John_i praises him_i
 - John_i praises himself_i
 - John_i thinks he_i is smart
 - *He_i thinks John_i is smart
 - *John_i thinks himself_i is smart
 - *John_i thinks that Mary likes himself_i
 - John_i considers himself_i to be intelligent
 - *John_i considers him_i to be intelligent

*John thinks himself is smart

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge C & T
9. merge V & C
10. merge v* & V
11. LR move to v*
12. merge T & v
13. move to spec-T
14. merge C & T

Last Resort (LR):
? theta merge

*John thinks himself is smart

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge C & T
9. merge V & C
10. merge v* & V
11. LR move to v*
12. merge T & v
13. move to spec-T
14. merge C & T

Probe [t] agrees with goal [d[d self][n[n he][d!case!arg[d][n john]]]]

T values Nominative Case of head *he*
Lexical gap: **heself*

Introduction: Data

- Classic Binding Theory data:
 - *John_i praises him_i
 - John_i praises himself_i
 - John_i thinks he_i is smart
 - *He_i thinks John_i is smart
 - *John_i thinks himself_i is smart
 - *John_i thinks that Mary likes himself_i
 - John_i considers himself_i to be intelligent
 - *John_i considers him_i to be intelligent

*John thinks that Mary likes himself

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) [] (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge V & N
5. merge v* & V
6. theta merge N & v
7. merge T & v
8. move to spec-T
9. merge C & T
10. merge V & C
11. merge v* & V

d (headed by *self*) is a phase boundary

Last Resort (LR):
theta merge

Probe [v*] agrees with goal [d[d self][n[n he][d!case!arg[d][n john]]]]

*John thinks that Mary likes himself

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge V & N
5. merge v* & V
6. theta merge N & v
7. merge T & v
8. move to spec-T
9. merge C & T
10. merge V & C
11. merge v* & V

Preference: Merge over Move (Chomsky 2000)
LR move of *John* blocked
derivation crashes

Introduction: Data

- Classic Binding Theory data:

*John_i praises him_i

John_i praises himself_i

John_i thinks he_i is smart

*He_i thinks John_i is smart

*John_i thinks himself_i is smart

*John_i thinks that Mary likes himself_i

John_i considers himself_i to be intelligent

*John_i considers him_i to be intelligent

Exceptional
Case Marking (ECM)

John considers himself to be intelligent

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) [] (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge V (ecm) & Tdef
9. merge v* & V
10. LR move to v*
11. merge T & v
12. move to spec-T
13. merge C & T

Infinitival clause
defective T
can't value Nominative Case

Probe [tdef] agrees with goal [d!case[d self][n[n he][d!case!arg[d][n john]]]

Scoreboard	<i>he</i>	<i>John</i>
needs	Case	theta, Case
licensed	theta	

John considers himself to be intelligent

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge V (ecm) & Tdef
9. merge v* & V
10. LR move to v*
11. merge T & v
12. move to spec-T
13. merge C & T

Scoreboard	<i>he</i>	<i>John</i>
needs	Case	theta, Case
licensed	theta	

John considers himself to be intelligent

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) [] (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge V (ecm) & Tdef
9. merge v* & V
10. LR move to v*
11. merge T & v
12. move to spec-T
13. merge C & T

Last Resort (LR):
theta merge applies

Probe [v*] agrees with goal [d[d self][n[n he][d!case!arg[d][n john]]]]

Scoreboard	<i>he</i>	<i>John</i>
needs		theta, Case
licensed	theta, Case	

John considers himself to be intelligent

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) [slider] (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge V (ecm) & Tdef
9. merge v* & V
10. LR move to v*
11. merge T & v
12. move to spec-T
13. merge C & T

Scoreboard	<i>he</i>	<i>John</i>
needs		Case
licensed	theta, Case	theta

John considers himself to be intelligent

TREE VIEWER

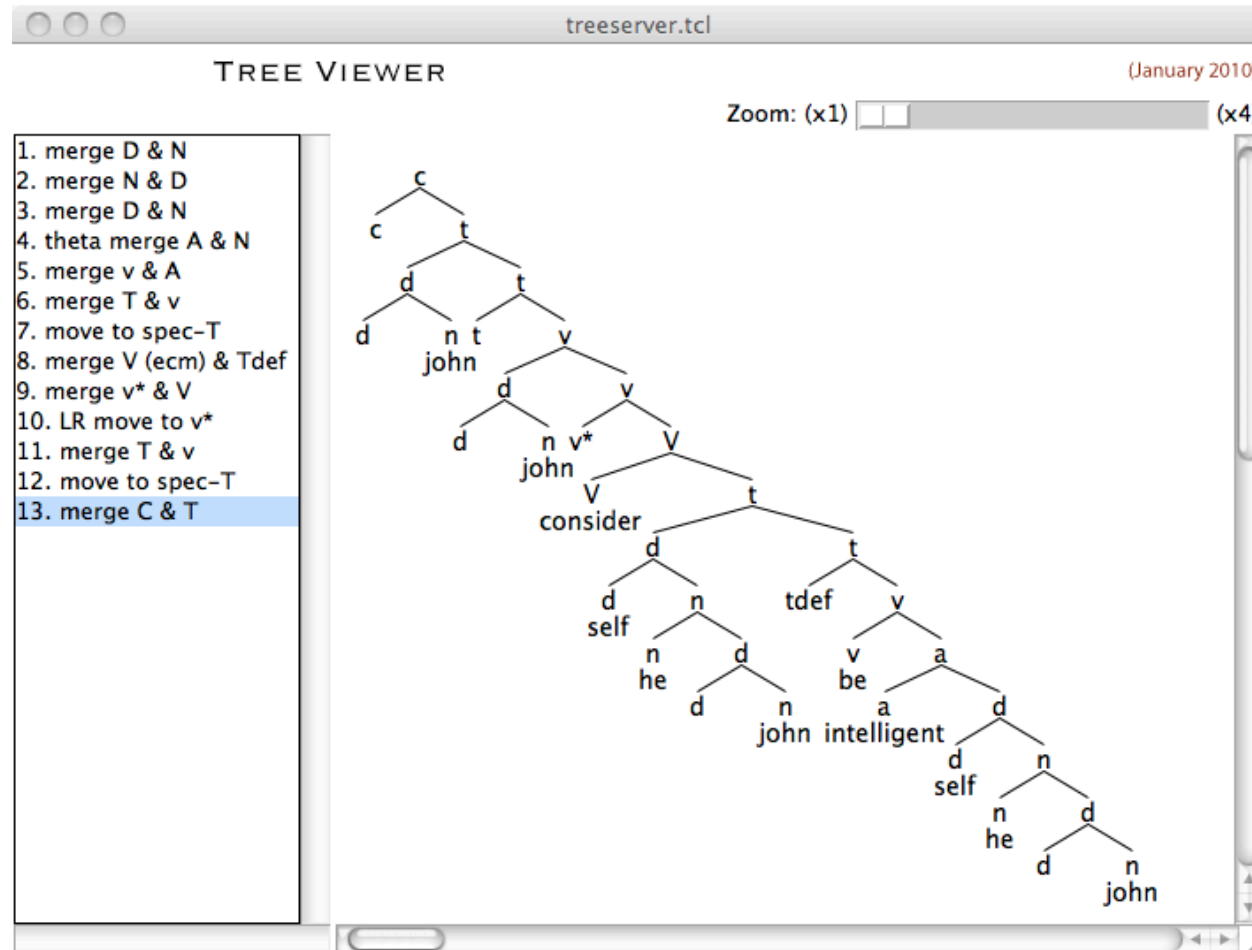
1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge V (ecm) & Tdef
9. merge v* & V
10. LR move to v*
11. merge T & v
12. move to spec-T
13. merge C & T

The diagram shows a syntax tree for the sentence "John considers himself to be intelligent". The root node is *t*, which branches into *t* and *v*. The left *t* branches into *d* (John) and *v**. The right *v* branches into *V* and *t*. The *V* node branches into *V* (consider) and *t*. The *t* node under *V* branches into *d* (self) and *n* (he). The *t* node under *V* branches into *tdef* and *v*. The *v* node branches into *a* (be) and *a*. The *a* node branches into *d* (John) and *n* (intelligent). The *a* node branches into *d* (self) and *n* (he). The *n* node branches into *d* (John) and *n* (John).

Scoreboard	<i>he</i>	<i>John</i>
needs		
licensed	theta, Case	theta, Case

Probe [t] agrees with goal [d[d][n john]]

John considers himself to be intelligent



*John_i considers him_i to be intelligent

treeviewer

TREE VIEWER

1. merge D & N
2. merge N & D
3. merge D & N
4. theta merge A & N
5. merge v & A
6. merge T & v
7. move to spec-T
8. merge V (ecm) & Tdef
9. merge v* & V

Recall, only way r-expr *John* can extract out of DC is using **Last Resort (LR)**: theta merge derivation stops (*crashes*)

Probe [v*] agrees with goal [d[d][n[n he][d!case!arg[d][n john]]]]

Introduction: Data

- Classic Binding Theory data:

John_i likes his_i dog

*John_i likes himself_i's dog

?*Hannah_i found a picture of her_i

Hannah_i found a picture of herself_i

?*Hannah found Peter_i's picture of him_i

Hannah found Peter_i's picture of himself_i

Hannah_i found Peter's picture of her_i

Hannah_i found Peter's picture of herself_i

Note: *picture*-DP
judgments from
Keller & Asudeh's
(2001) study

John_i likes his_i dog

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. merge D & N
5. theta merge D & D
6. theta merge V & N
7. merge v* & V
8. LR move to v*
9. merge T & v
10. move to spec-T
11. merge C & T

Assume *his* = *he* + 's
 [[*he John*][*'s [dog]*]]

d (possessive) is a phase boundary

```

graph TD
    v_star((v*)) --- V((V))
    v_star --- v_star_child((v*))
    V --- like((like))
    V --- d1((d))
    d1 --- d2((d))
    d1 --- d3((d))
    d2 --- he((he))
    d2 --- n1((n))
    n1 --- john((john))
    d3 --- s(('s))
    d3 --- dog((dog))
    
```

Last Resort (LR):
theta merge applies

Probe [v*] agrees with goal [d[d!case[d][n[n he][d!case!arg[d][n john]]]][d[d 's][n dog]]]

John_i likes his_i dog

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. merge D & N
5. theta merge D & D
6. theta merge V & N
7. merge v* & V
8. LR move to v*
9. merge T & v
10. move to spec-T
11. merge C & T

d (possessive) is a phase boundary

John_i likes his_i dog

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. merge D & N
5. theta merge D & D
6. theta merge V & N
7. merge v* & V
8. LR move to v*
9. merge T & v
10. move to spec-T
11. merge C & T

```
graph TD
    c1[c] --- c2[c]
    c1 --- t1[t]
    c2 --- d1[d]
    c2 --- n1[n]
    n1 --- john1[john]
    t1 --- t2[t]
    t1 --- v1[v]
    t2 --- d2[d]
    t2 --- n2[n]
    n2 --- john2[john]
    v1 --- vstar[v*]
    v1 --- V1[V]
    vstar --- d3[d]
    vstar --- n3[n]
    n3 --- john3[john]
    V1 --- V2[V]
    V1 --- d4[d]
    V2 --- V3[V]
    V2 --- d5[d]
    V3 --- like[like]
    d4 --- d6[d]
    d4 --- n4[n]
    n4 --- he[he]
    d5 --- d7[d]
    d5 --- n5[n]
    n5 --- john4[john]
    d6 --- d8[d]
    d6 --- n6[n]
    n6 --- dog[dog]
    d7 --- d9[d]
    d7 --- n7[n]
    n7 --- s[s]
```


*John likes himself's dog

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. merge D & N
2. merge N & D
3. merge D & N
4. merge D & N
5. theta merge D & D
6. theta merge V & N
7. merge v* & V
8. merge D & N
9. merge N & D
10. merge D & N
11. theta merge V & N
12. merge v* & V
13. LR move to v*
14. merge T & v
15. move to spec-T
16. merge C & T

LR move of *John* to subject position blocked

Probe [v*] agrees with goal [d[d!case[d self][n[n he][d!case!arg[d][n john]]]][d[d 's][n dog]]]

actually, there's a more subtle subcase of LR move blocking that needs to be considered when *self-he-John* is merged with *'s-dog*
LR move blocked by Merge over Move

*John likes himself's dog

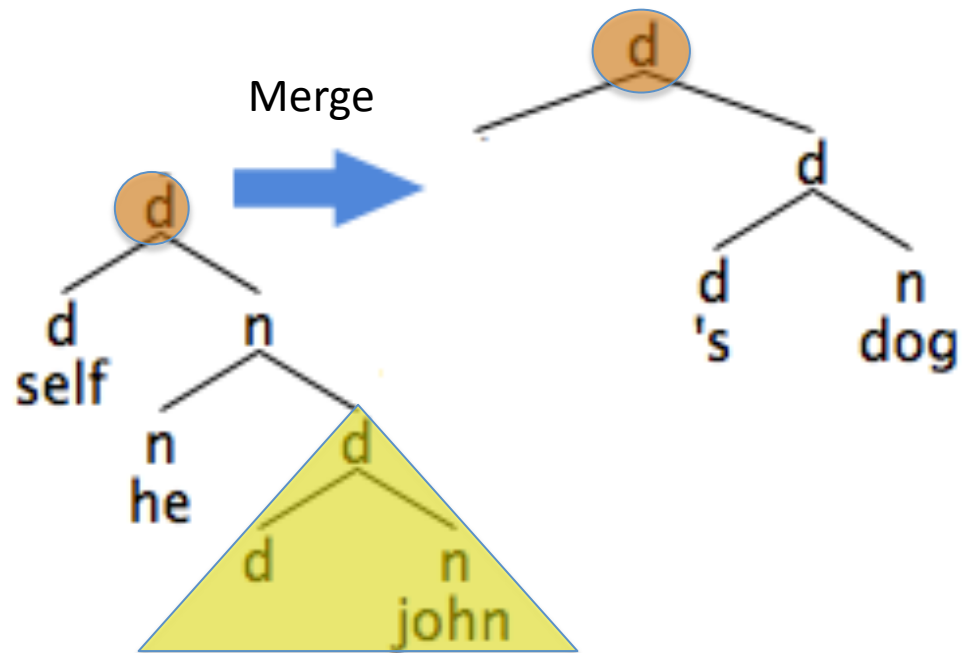
Why didn't *John* get a chance to move earlier?

Two ways to think about it

1. LR move to spec-D blocked by Merge over Move
2. Internal merge (move) can only apply to already merged syntactic objects.

LR move is (still) an instance of internal merge

Once *self-he-John* merges with 's-dog, it's too late



Introduction: Data

- Classic Binding Theory data:

John_i likes his_i dog

*John_i likes himself_i's dog

?*Hannah_i found a picture of her_i

Hannah_i found a picture of herself_i

?*Hannah found Peter_i's picture of him_i

Hannah found Peter_i's picture of himself_i

Hannah_i found Peter's picture of her_i

Hannah_i found Peter's picture of herself_i

Note: *picture*-DP
judgments from
Keller & Asudeh's
(2001) study

Hannah found Peter's picture of her

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. theta merge V & N
2. merge v* & V
3. merge D & N
4. theta merge N & D
5. theta merge V & N
6. merge v* & V
7. merge D & N
8. merge N & D
9. merge D & N
10. theta merge P & N
11. merge N & P
12. merge D & N
13. theta merge N & D
14. theta merge V & N
15. merge v* & V
16. LR move to v*
17. merge T & v
18. move to spec-T
19. merge C & T

```
graph TD
    v((v)) --- d1((d))
    v --- V((V))
    d1 --- hannah1[hannah]
    V --- v_star((v*))
    V --- d2((d))
    v_star --- find[find]
    d2 --- d3((d))
    d2 --- n1((n))
    d3 --- peter[peter]
    n1 --- apostrophe['s']
    n1 --- n2((n))
    n2 --- picture[picture]
    n2 --- p((p))
    p --- of[of]
    p --- d4((d))
    d4 --- n3((n))
    d4 --- n4((n))
    n3 --- she[she]
    n4 --- d5((d))
    d5 --- hannah2[hannah]
```

Hannah found Peter's picture of her

treeviewer.tcl

TREE VIEWER (January 2010)

Zoom: (x1) (x4)

1. theta merge V & N
2. merge v* & V
3. merge D & N
4. theta merge N & D
5. theta merge V & N
6. merge v* & V
7. merge D & N
8. merge N & D
9. merge D & N
10. theta merge P & N
11. merge N & P
12. merge D & N
13. theta merge N & D
14. theta merge V & N
15. merge v* & V
16. LR move to v*
17. merge T & v
18. move to spec-T
19. merge C & T

```
graph TD
    c[c] --- d1[d]
    c --- t[t]
    d1 --- hannah1[hannah]
    t --- n1[n]
    t --- v1[v]
    n1 --- hannah2[hannah]
    v1 --- d2[d]
    v1 --- v2[v]
    d2 --- hannah3[hannah]
    v2 --- V[V]
    v2 --- d3[d]
    V --- find[find]
    d3 --- d4[d]
    d3 --- n2[n]
    d4 --- peter[peter]
    n2 --- apostrophe['s]
    n2 --- n3[n]
    n3 --- picture[picture]
    n3 --- p[p]
    p --- of[of]
    p --- d5[d]
    d5 --- n4[n]
    d5 --- d6[d]
    n4 --- she[she]
    d6 --- d7[d]
    d6 --- n5[n]
    d7 --- hannah4[hannah]
```

Hannah found Peter's picture of herself

treeviewer.tcl
TREE VIEWER
(January 2010)
Zoom: (x1) (x4)

1. theta merge V & N
2. merge v* & V
3. theta merge N & D
4. theta merge V & N
5. merge v* & V
6. merge D & N
7. merge N & D
8. merge D & N
9. theta merge P & N
10. merge N & P
11. theta merge N & D
12. theta merge V & N
13. merge v* & V

judgment from Keller & Asudeh(2001)

LR move of *Hannah* to subject position blocked

Probe [v*] agrees with goal [d[d!case[d][n peter]][d[d 's][n[n picture][p[p of][d[d self][n[n she][d!case!arg[d][n hannah]]]]]]]]]

Other data

To-do list ...

- **transitivity of coreference**
 - John_i says he_i thinks he_i's smart (Kayne 2002:157)
- **circularity**
 - [His_i wife]_j just saw [her_j husband]_i (Kayne 2002:156)
- **wh-constructions**
 - Which book that John_i wrote does he_i like best (Zwart 2002)
 - Who that John_i knows does he_i admire? (Chomsky & Lasnik 1995:106)
 - Which stories about Diana_i did she_i most object to? (Zwart 2002)
 - *How many stories about Diana_i does she_i want us to invent. ((Heycock 1995:558f), per Zwart 2002)
- **anaphors and pronouns in non-complementary distribution**
 - Lucie counted five tourists in the room apart from herself_i/her_i (Reinhart & Reuland 1993:661)
 - Lucie saw a picture of herself_i/her_i (Reinhart & Reuland 1993:661)
 - Max likes jokes about himself_i/him_i (Reinhart & Reuland 1993:661)

Summary

- computational implementation *theory verification*

- accounts for bunch of classic Binding Theory facts *sans Conditions A, B and C*

- theory:

basic MP (Chomsky 2000)

+ doubling constituent (Kayne 2002)

+ some DPs are phases

+ LR move

extra mechanism... yes

But doesn't impact efficient computation

e.g. creates no extra choice points

LR theta merge: competes with external Merge

also Chomsky (2006),
Svenonious (2004),
Hiraiwa (2005)