Minimalist Parsing: simplicity and feature unification

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

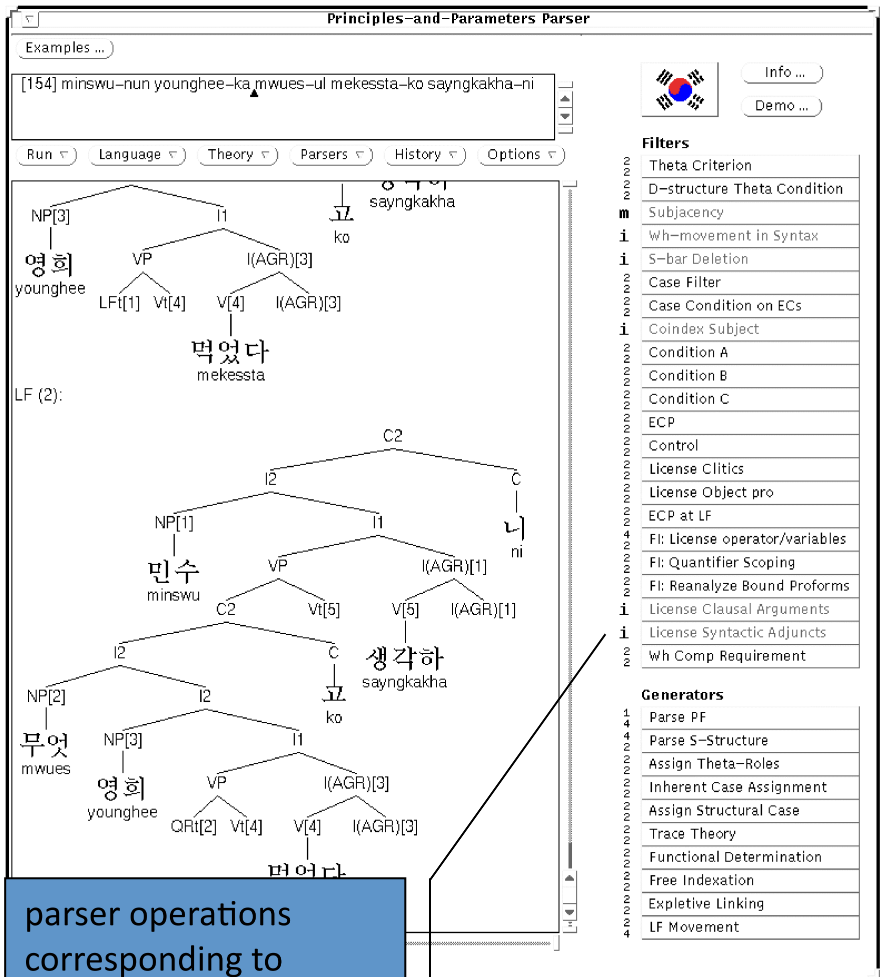
Conference on Language and Recursion. University of Mons, Belgium. March 14-16th 2011

Linguistic Theory

Computation and Linguistic Theory

- Principles and Parameters Approach (Chomsky 1980)
  - Linguistic principle = declarative statement:
    - Noun phrases must receive (abstract) Case (*Case Filter*)
    - An anaphor must be A-bound in its Governing Category (*Binding Principle A*)
- Minimalist Program (e.g. Chomsky 2001)
  - Precisely specified sequence of (feature-driven) operations
    - Merge: external and internal (= Move)
    - Agree: Probe and Goal
      (interaction between uninterpretable and interpretable features)

Principles and Parameters Approach



parser operations corresponding to linguistic principles

PAPPI: Fong (1990)

Theory is (relatively) unconstrained

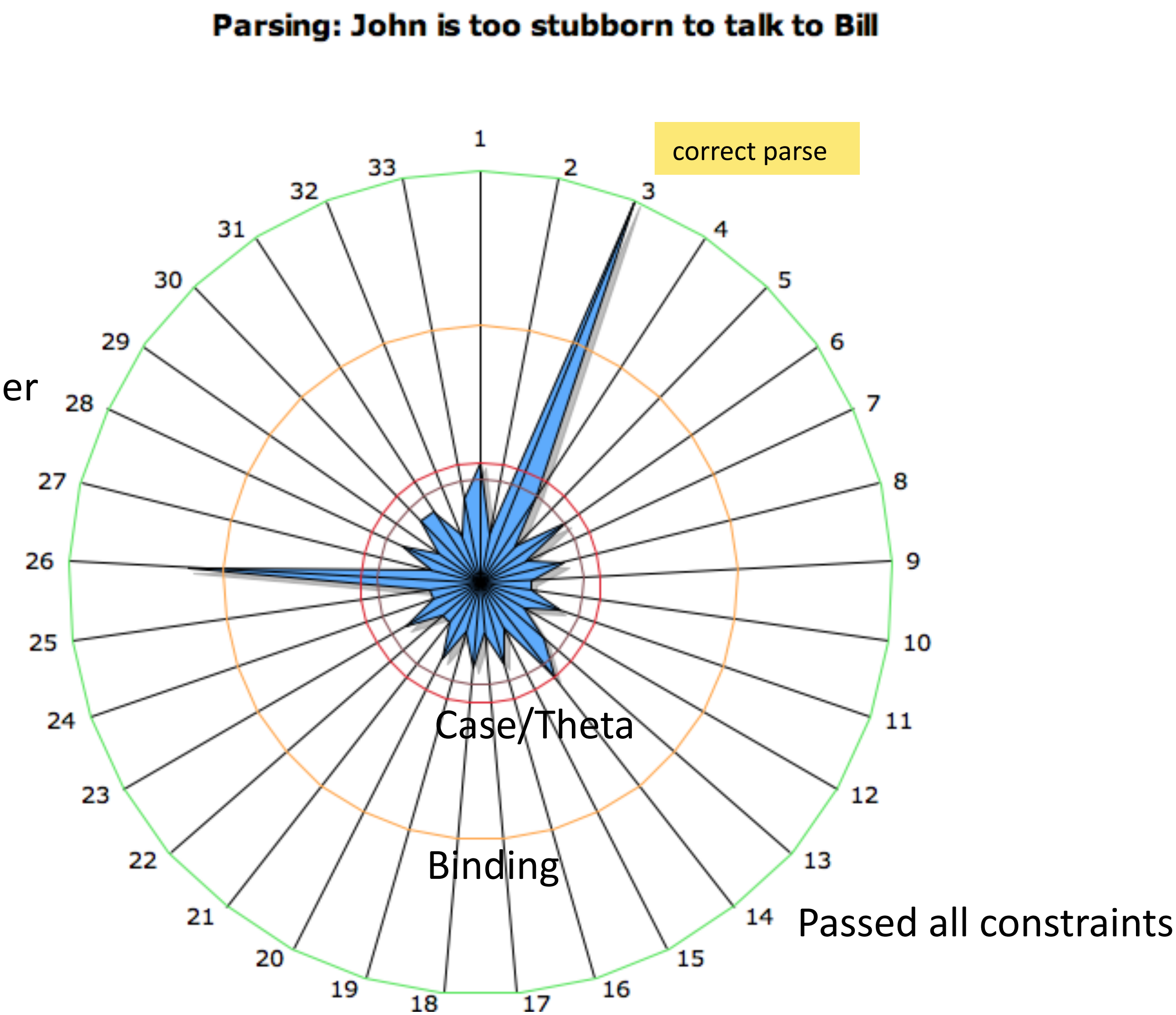
- declarative principles can be “coded up” and combined in many different ways and yet remain “faithful” to the theory

Principles and Parameters Approach

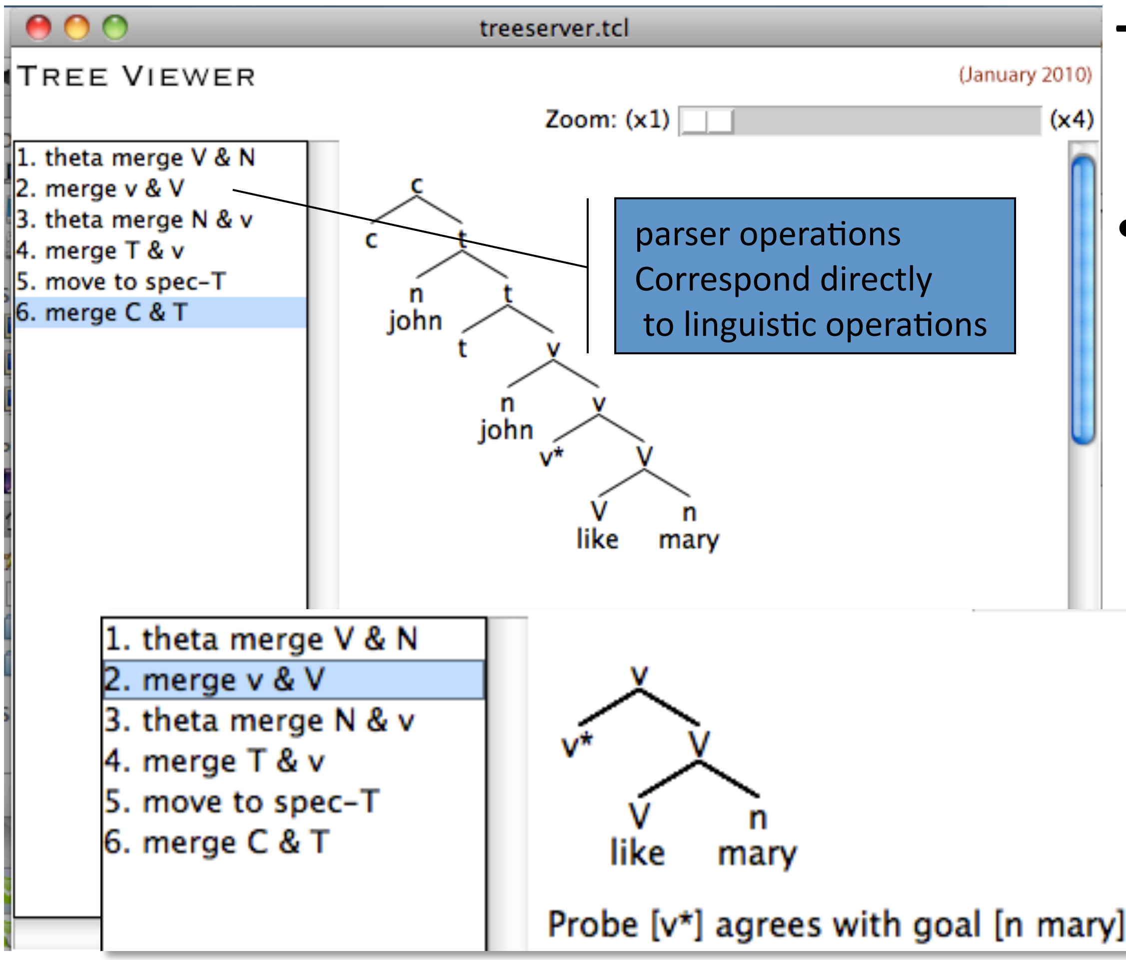
Generate and test:

33 candidate parses examined by the parser

most are eliminated early by Case/Theta theory sub-system



Minimalist Program



Theory is highly constrained

- at each Merge step, operations are precisely specified, not much “wriggle room” for implementation

Prior work on grammar formalisms for MP theories e.g.
(Stabler, 1998)
(Lecomte & Retoré, 2001)

A Simple Implementation

- Take a theory in the Minimalist Program
  - e.g. Derivation by Phase (Chomsky 2001)

- What is the simplest possible computational implementation that we can get away with?

Simple does not necessarily equal efficient or "minimal"

Example

- Task: sorting a list of numbers (*n*: size of list)
  - 5
  - 3
  - 8
  - 1
  - 7
  - 9
  - 4 …

- Simplest implementation
  - only operation is to front lowest number
  - $O(n^2)$ comparisons

- Quicksort
  - recursively sort around a ***pivot*** number
  - same worst case but typically fast with $O(n \lg n)$ comparisons

Scaling is important in sorting but for linguistic computation?

Example

- Phrase Structure Grammar
  - S ⇨ NP VP
  - VP ⇨ V NP
  - VP ⇨ V
  - VP ⇨ VP PP
  - NP ⇨ D N
  - NP ⇨ N
  - PP ⇨ P NP

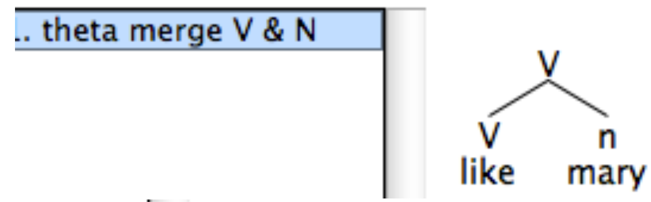
- Use the grammar directly
  - recursive descent
  - bottom-up

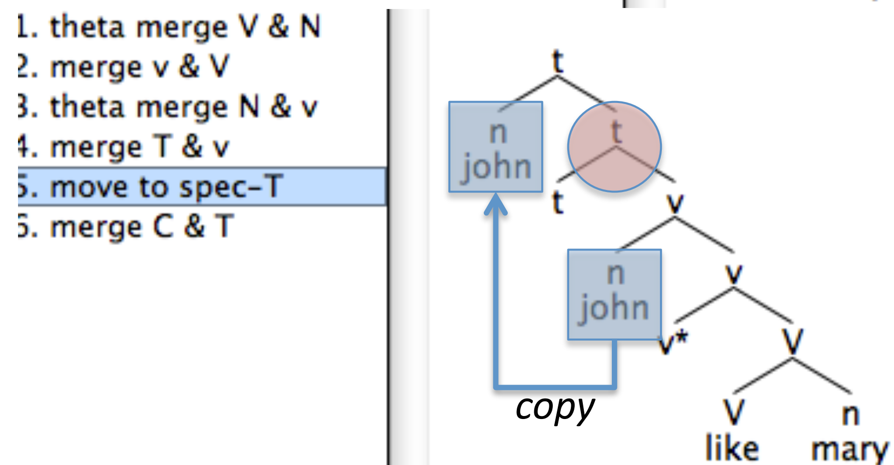
- Transform the grammar into a finite state machine with a stack
  - Earley's algorithm (*as you go*)
  - LR(k) Parsing (*offline is for free*)

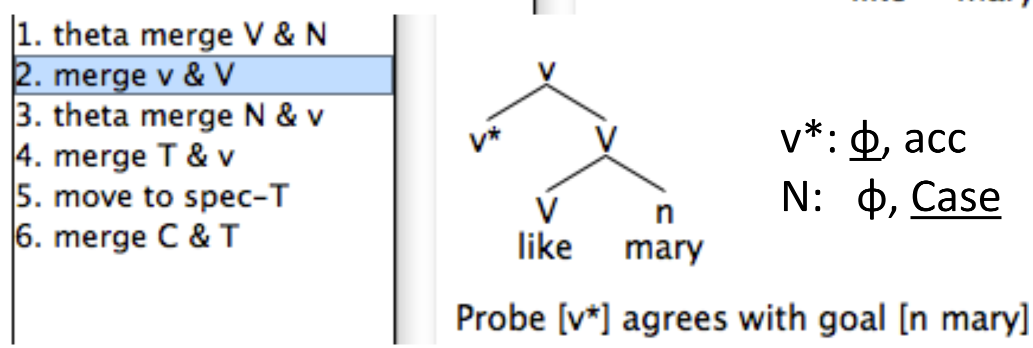
Search space: Minimize the length of the derivation

Basic Computation

- start with lexical array of syntactic objects: {α,..,ω}
- **Merge** *"an indispensible operation of a recursive system"*
- (external)
  - two syntactic objects (SOs): α, β
  - create merged SO: {α, β}
  - label({α, β})=label(α) or label(β)
- (internal), implements **Displacement**
  - SOs: α and γ, γ properly contained in α
  - create SO: {α, γ}
  - label({α, γ})=label(α)
- **Agree**
  - active probe SO: α (active = uninterpretable features), goal SO: β
  - match and delete uninterpretable features of probe and goal
- **Convergent derivation**: uninterpretable features must be eliminated



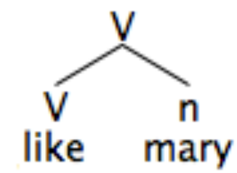




Basic Implementation (1)

Definite clause grammar (DCG) (*simplified*)

```
V([V V N])  --> V(V), n(N).
V([V Verb]) --> [Verb].
```



	features (V)	features (N)
– phonetic matrix:	f(pmatrix,*like*)	f(pmatrix,*john*)
– (takes an) argument:	f(arg,+)	f(arg,+)
– uninterpretable Case:		f(case,_)

variable

```
% (big V) verb classes
bV(n('V',[],[V,N])) --> bV0(V), n0(N), {theta(V),theta(N)} report 'theta merge V & N'.
bV0(n('V',[f(pmatrix,Verb),f(arg,+)],[])) --> [Verb], {transitive(Verb)}.
bV0(n('V',[f(pmatrix,Verb),f(arg,+)],[])) --> [Verb], {unaccusative(Verb)}.
bV0(n('V',[f(pmatrix,Verb)],[])) --> [Verb], {unergative(Verb)}.

transitive(like).     transitive(expect).
unergative(run).      unaccusative(arrive).

n0(n(n,[f(pmatrix,BNoun)|Fs],[])) --> [BNoun], {bareNoun(BNoun,Fs)}.
bareNoun(john,[f(phi,3-sg-m),f(case,_),f(arg,+)]).
```

Basic Implementation (2)

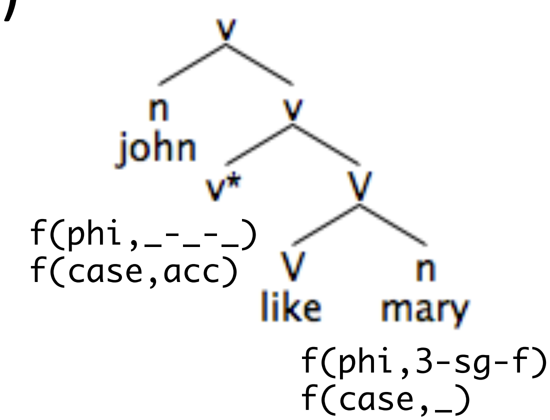
Definite clause grammar (DCG) (*simplified*)

v([$_{v}$ N v]) --> n(N), v(v).

v([$_{v}$ v V]) --> v(v), V(V).

v([$_{v*}$]) --> [].

– features (v*)
  - uninterpretable ϕ-features: f(phi,_-_-_)
  - can value accusative Case: f(case,acc)

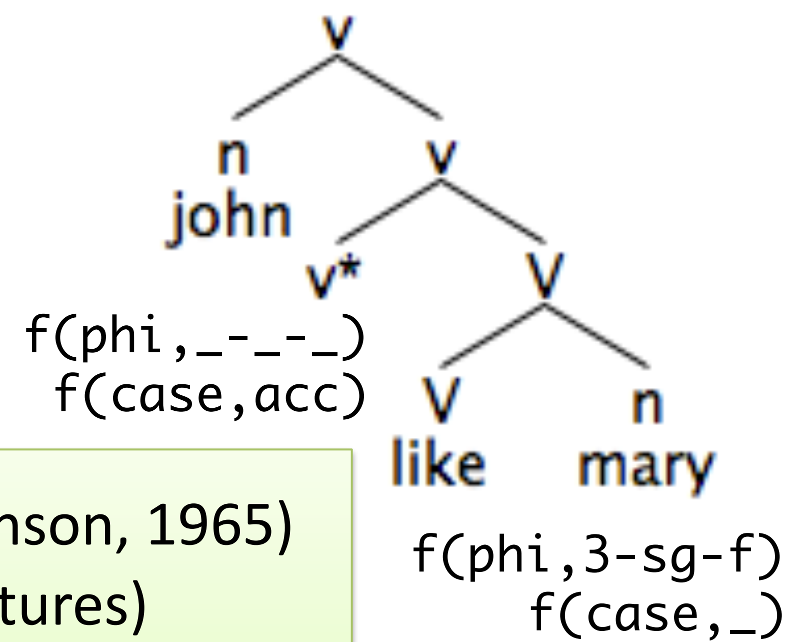


triggers Agree

```
% v*/vP
v(n(v,[],[N,V])) --> n0(N), {theta(N)}, v1(V) report 'theta merge N & v'.
v1(n(v,[],[V,BV])) --> v0(V), bV(BV), {goals(BV,Goals),agree(V,Goals)}
report 'merge v & V'.
v0(n('v*',[f(phi,_-_-_),f(case,acc)],[])) --> [].
```

Basic Implementation (2)

- Agree(v*,N)



Operation: **unification** (Robinson, 1965)
(match and instantiate unvalued features)

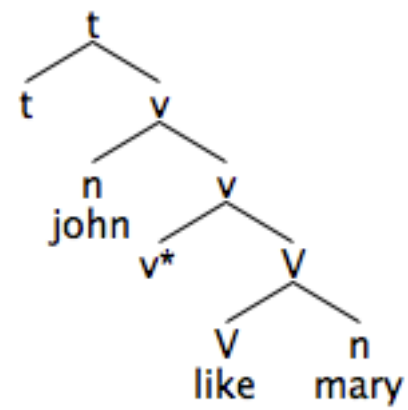
<u>Probe</u>	<u>Goal</u>
`f(phi,_-_-_)`	`f(phi,3-sg-f)`
`f(case,acc)`	`f(case,_)`

Uninterpretable/unvalued features (*represented by variables*) are eliminated

Basic Implementation (3)

Definite clause grammar (DCG)

- T([$_T$ N v]) --> n(N), v(v).
- T([$_T$ N v]) --> v(v), {*N a goal*}
- T([$_T$]) --> [].
- – features (ϕ-complete T)
  - uninterpretable ϕ-features: `f(phi,_-_-_)`
  - can value nominative Case: `f(case,nom)`
  - EPP



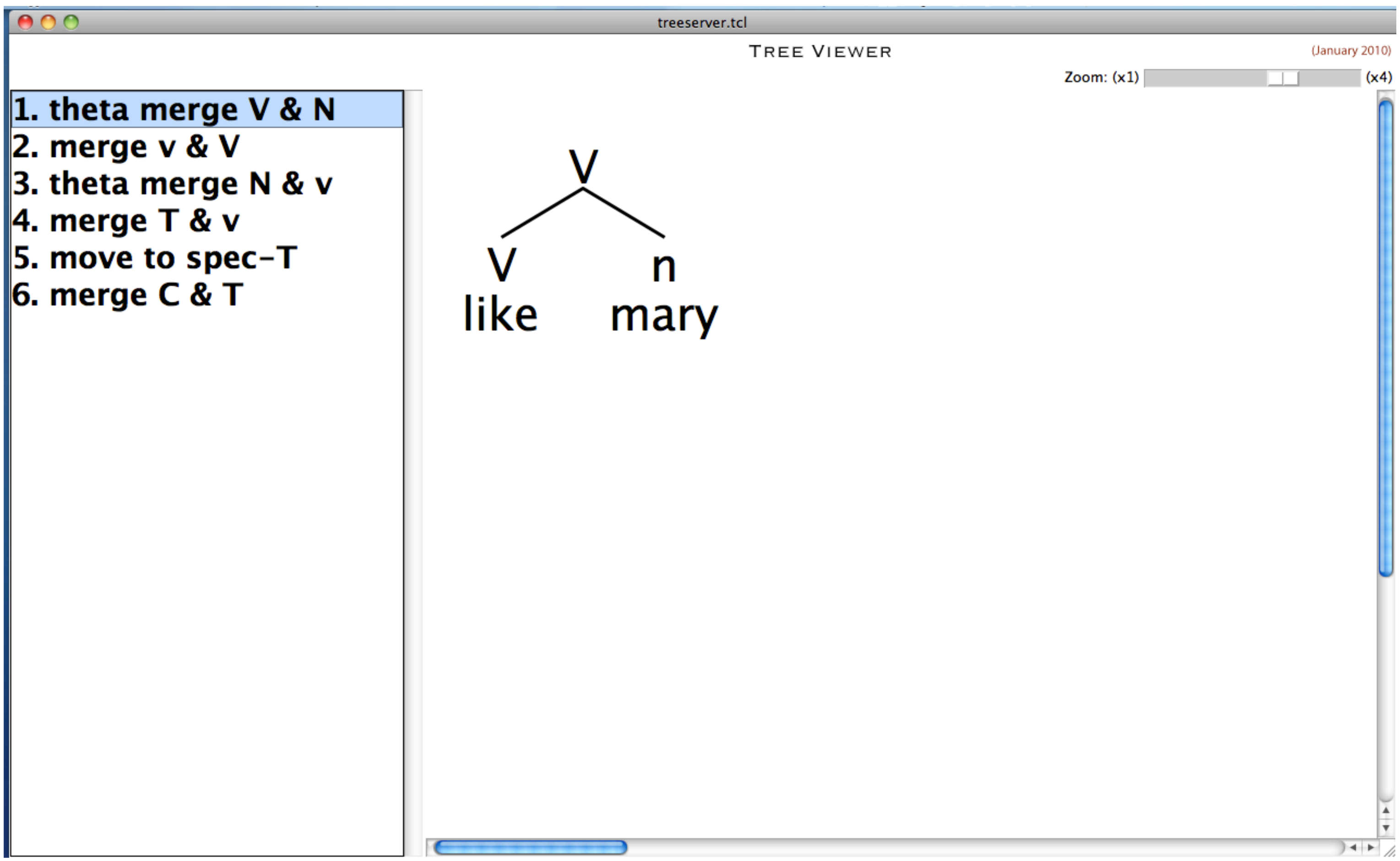
defective T
ϕ-incomplete
e.g. infinitivals

triggers Agree

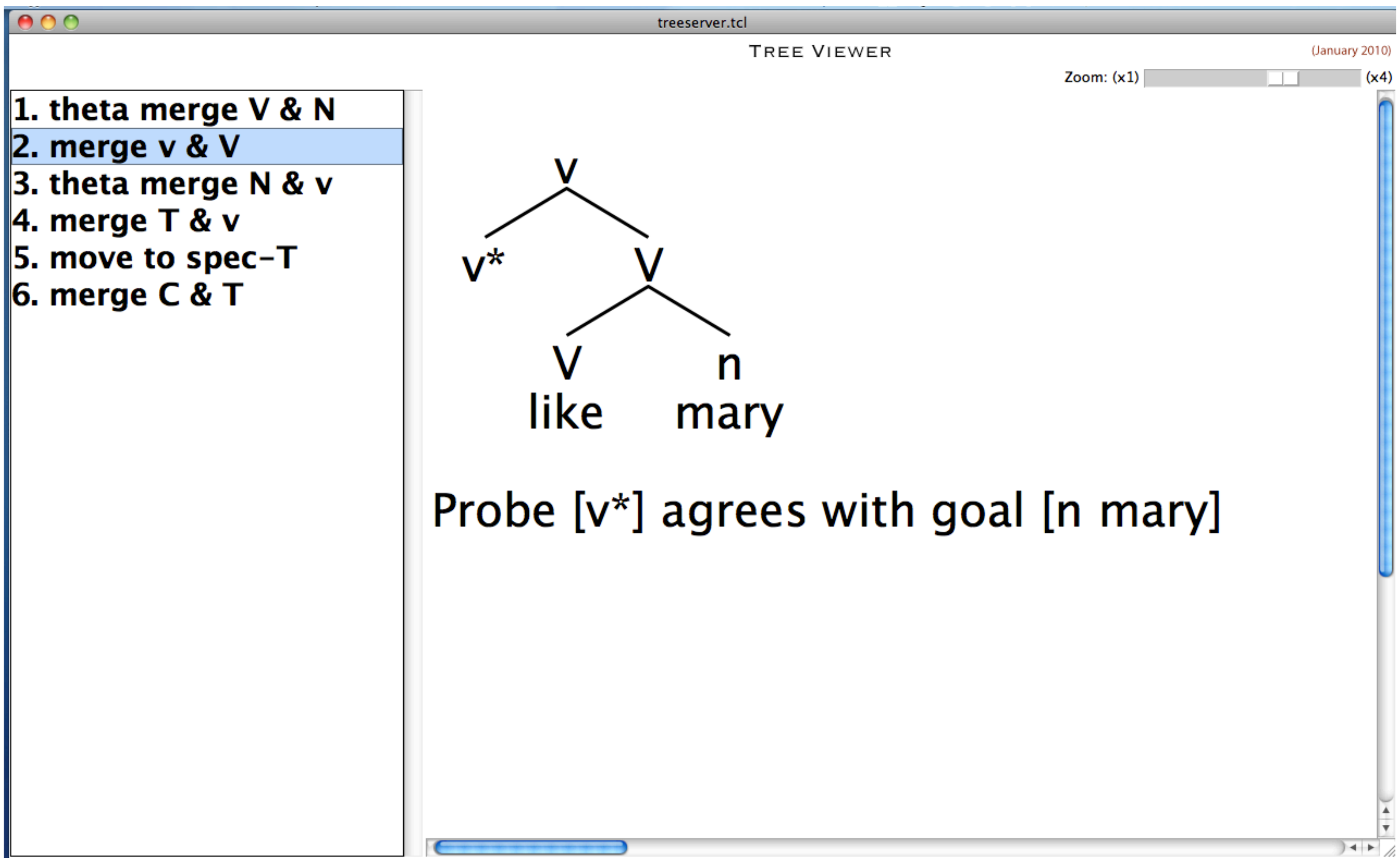
```
t(n(t,[],[N,T])) --> n0(N), {nonarg(N)}, t1(T,_) report 'merge expl & T'.
% EPP: (1) merge
t(n(t,[],[Goal,T])) --> t1(T,Goal) report 'move to spec-T'.
% EPP: (2) move with maximize matching

t1(n(t,[],[T,V]),G) --> t0(T), v(V), {goals(V,Goals),agree(T,Goals),
Goals = [G|_]} report 'merge T & v'.
t0(n(t,[f(phi,_-_-_),f(case,nom)],[])) --> [].
```

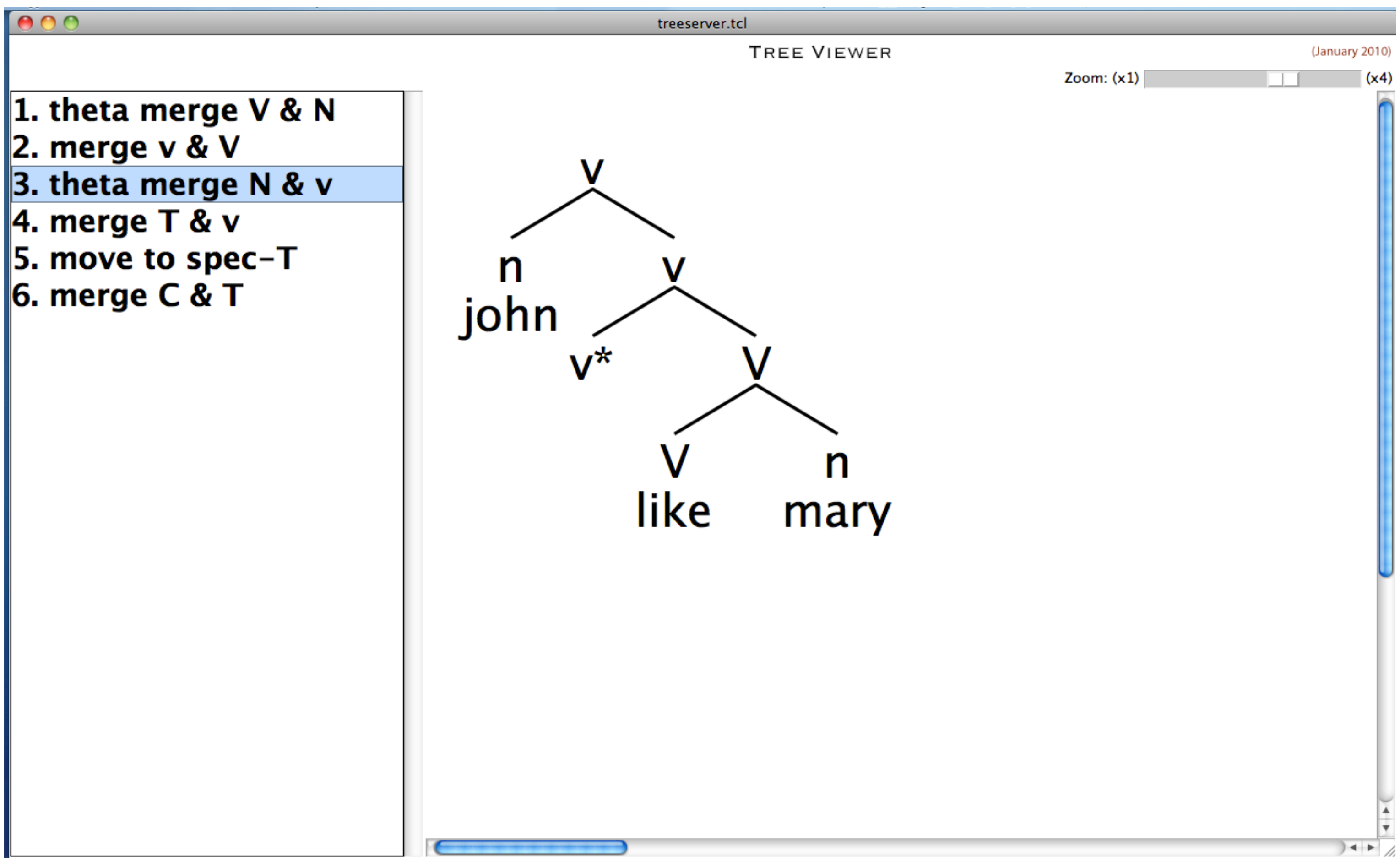
Putting it all together (1)



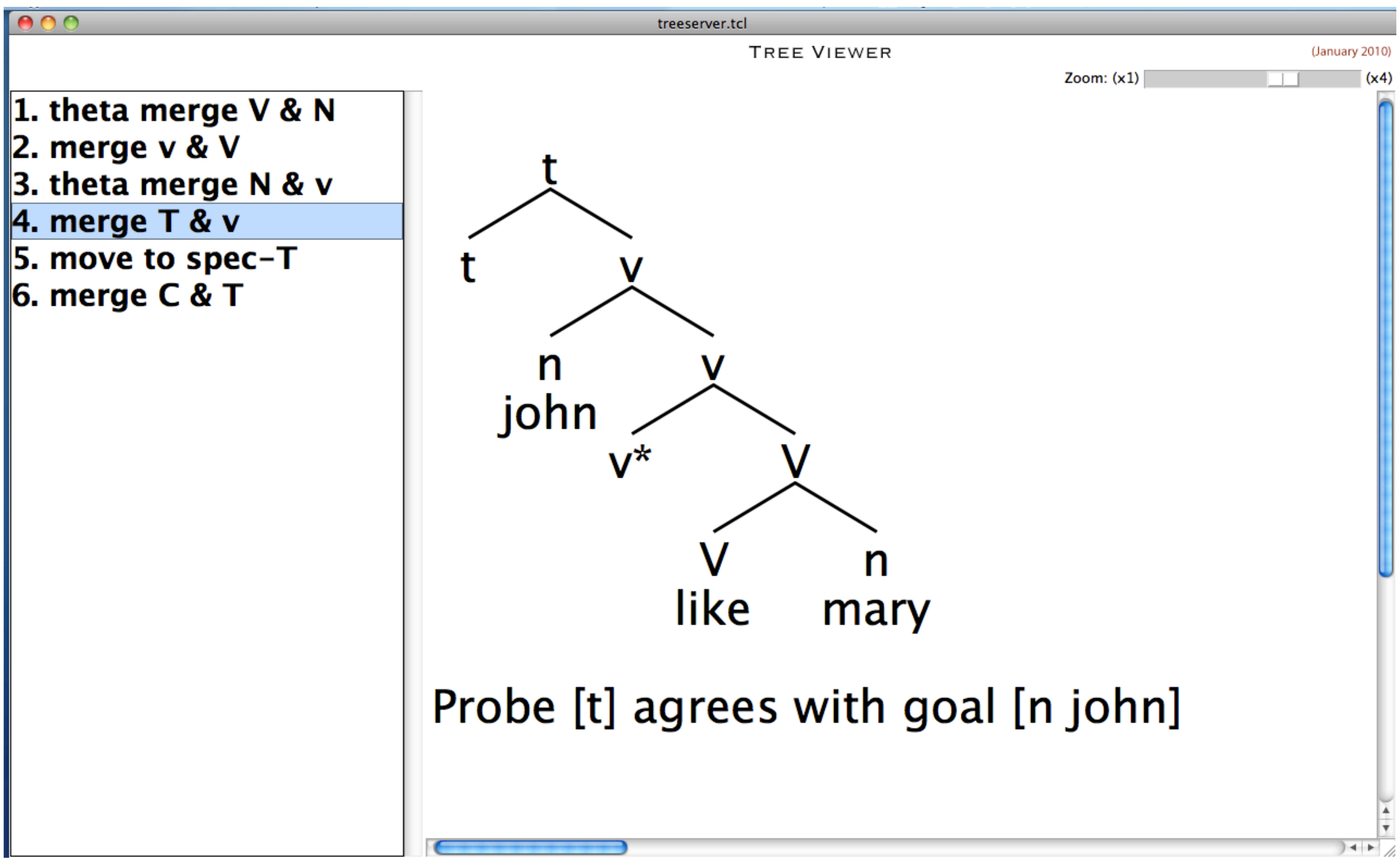
Putting it all together (2)



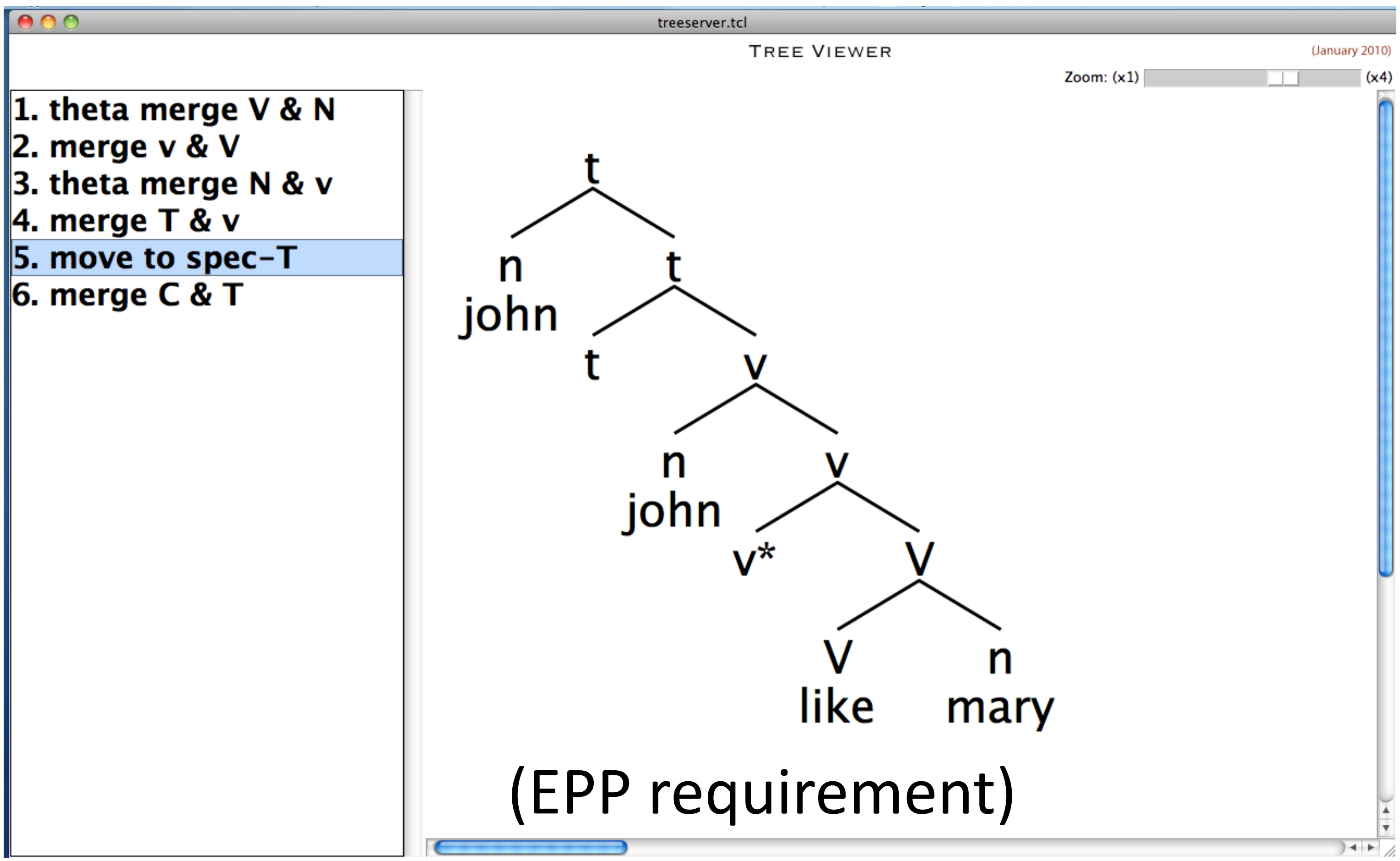
Putting it all together (3)



Putting it all together (4)

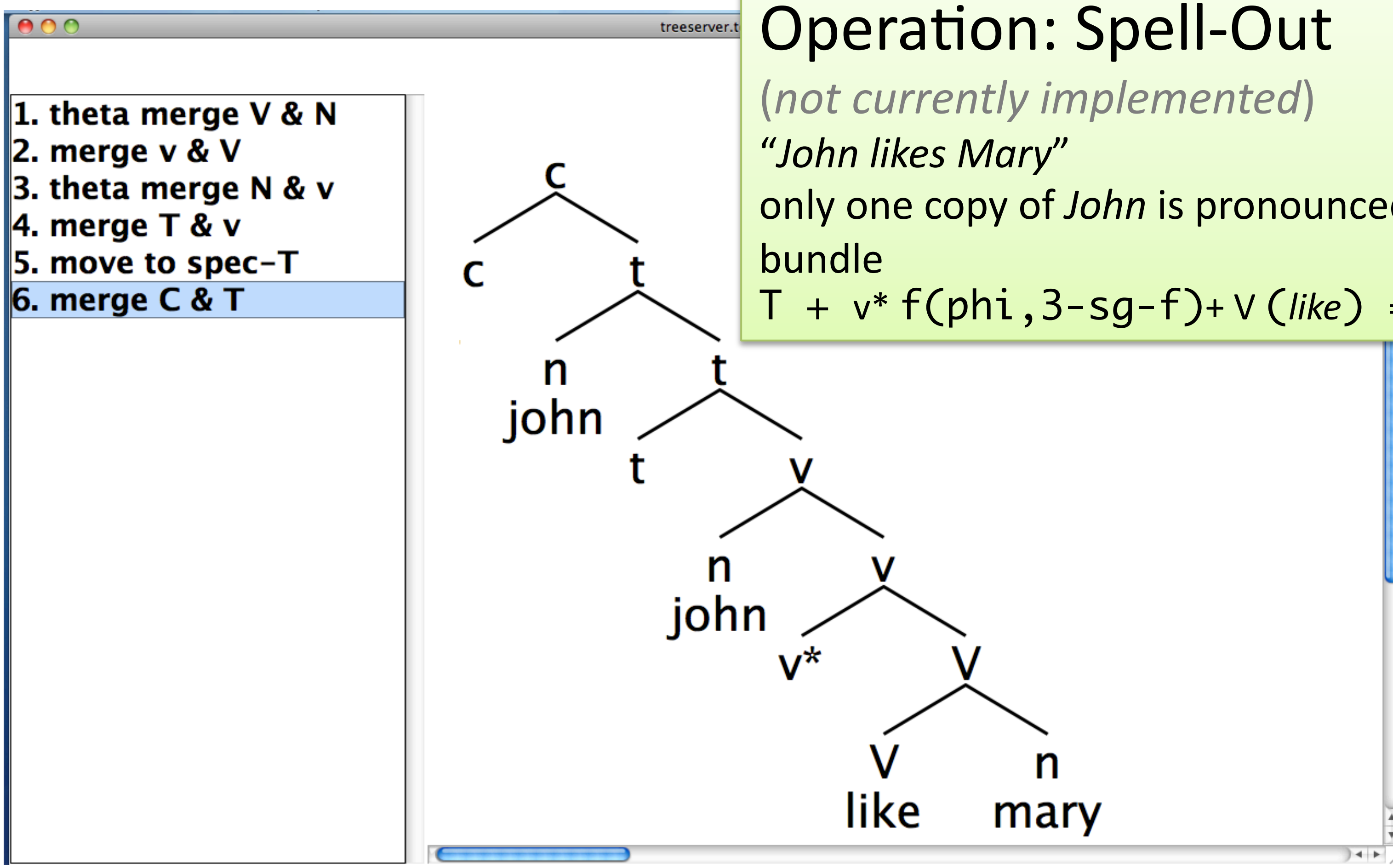


Putting it all together (5)



(EPP requirement)

Putting it all together (6)



Operation: Spell-Out

(*not currently implemented*)

"John likes Mary"

only one copy of *John* is pronounced

bundle

T + v* f(phi,3-sg-f)+ V (*like*) = *likes*

Examples

assume, are raising constructions and their exceptional-Case-marking (ECM) counterparts, as shown schematically in (4a), where β is the matrix clause, α is an infinitival with YP a verbal phrase (the case most relevant here), and P is the probe: T with a raising verb (case (4b)), *v* with an ECM transitive verb (case (4c)).[10]

(4) a. $[_\beta$ P $[_\alpha$ [Subj [H YP]]]]
  b. i. there are likely to be awarded several prizes
    ii. several prizes are likely to be awarded
  c. i. we expect there to be awarded several prizes
    ii. we expect several prizes to be awarded

The Case/agreement properties of Subj in (4a), and its overt location, are determined by properties of the matrix probe P, not internally to α. α is a TP with defective head T_{def}, which is unable to determine Case/agreement but has an EPP-feature, overtly manifested in (4c). Raising-ECM parallels give good reason to believe that the EPP-feature is manifested in (4b) as well, by trace of the matrix subject; preference for Merge over (more complex) Move gives a plausible reason for the surface distinction between [Spec, T_{def}] in (4b) and in (4c) (see MI). In (4bi) and (4ci), the EPP-feature of T_{def} is satisfied by Merge of expletive; in (4bii) and (4cii), by raising of the direct object.

(Chomsky 2001)

A Worked Example

Consider the derivation of

- several prizes are likely to be awarded (= 4(b)(ii))

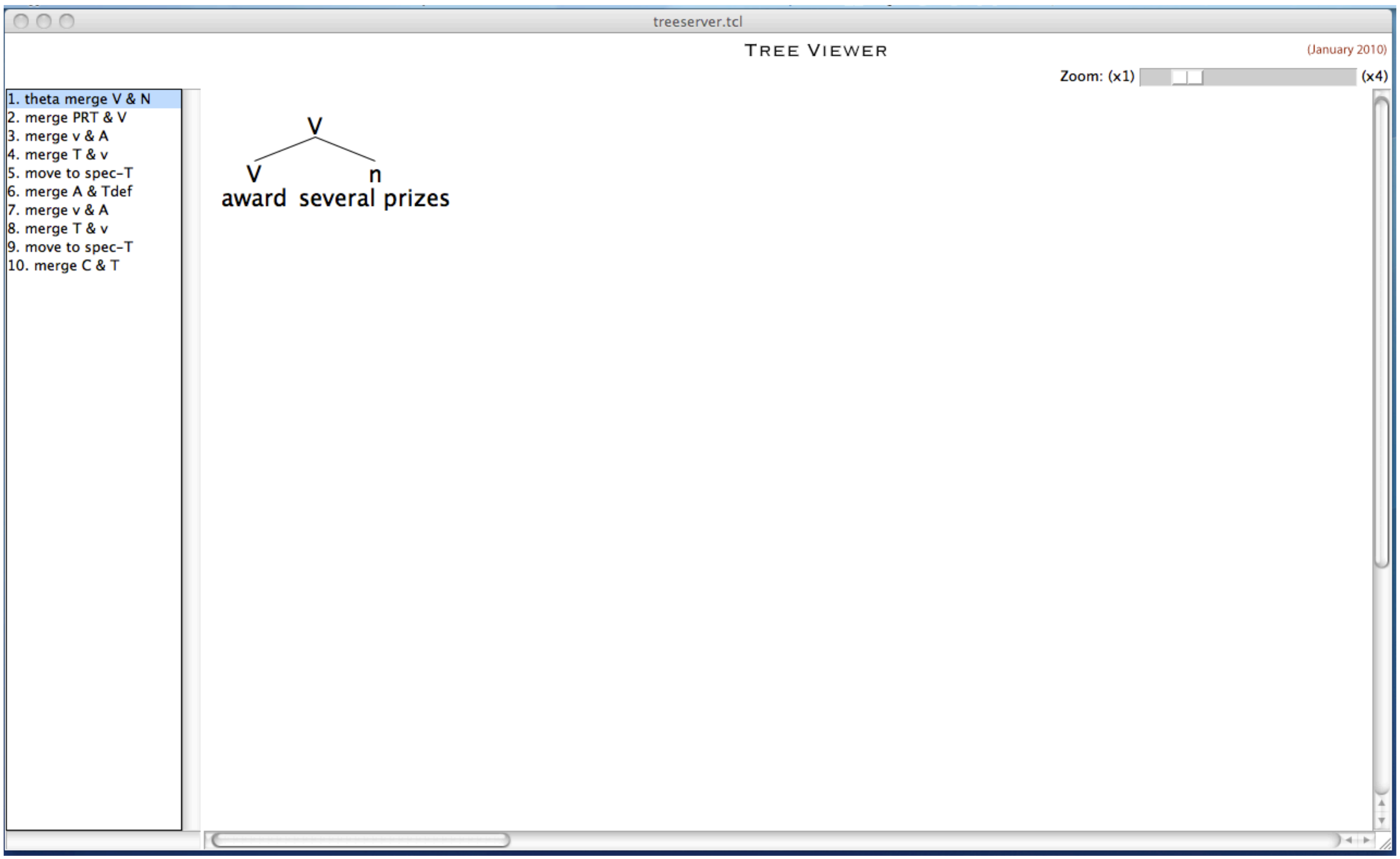
  awarded = *award* + *-ed* (adjectival participle)

  -ed ϕ−incomplete: uninterpretable Number and Gender only
  uninterpretable Case
  morphologically unrealized in English (cf. Icelandic)

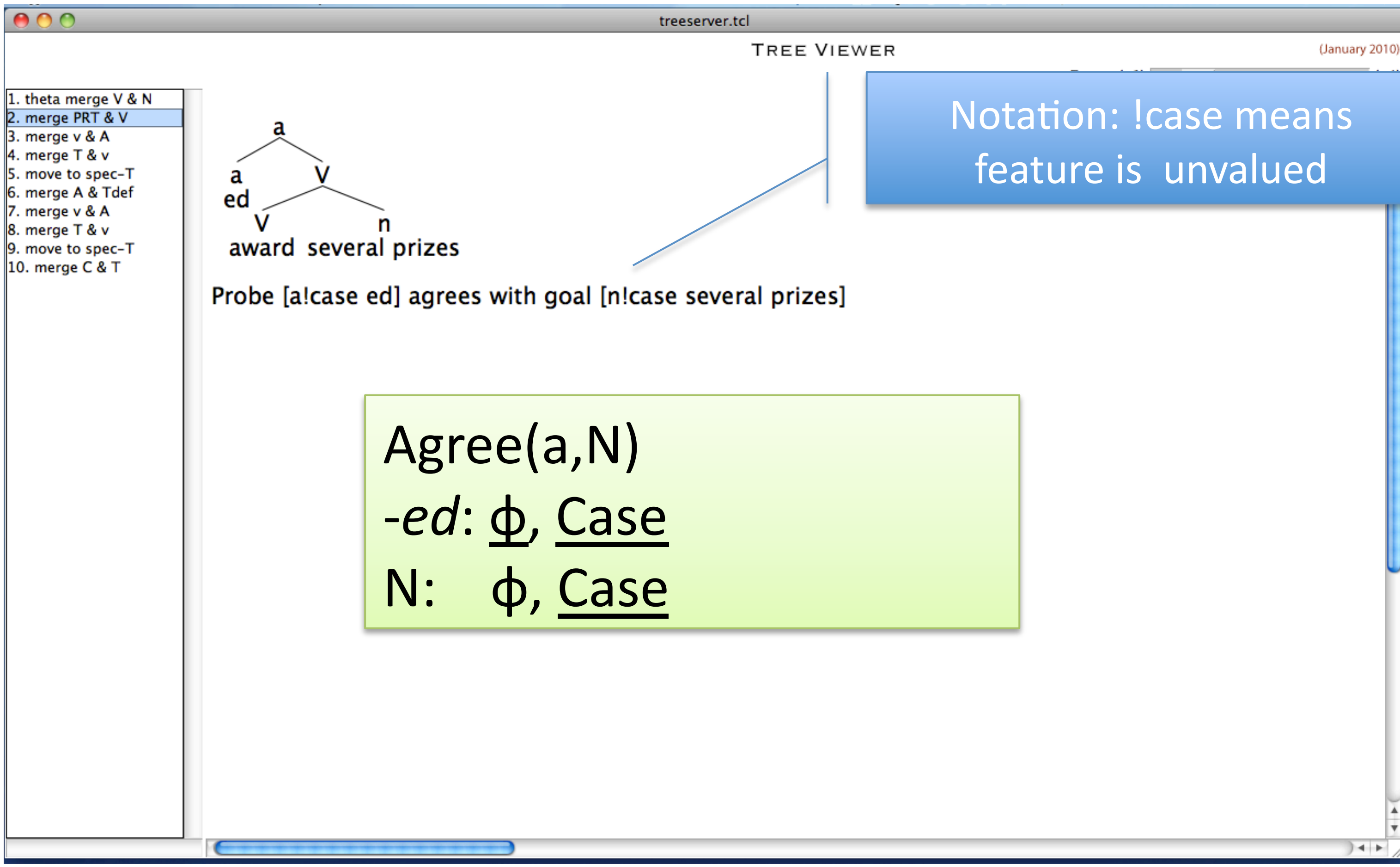
Agree(a,N)
-ed: <u>ϕ</u>, <u>Case</u>
N: ϕ, <u>Case</u>

```
?- parse([be,likely,be,ed,award,several,prizes]).
Probe [a!case ed] agrees with goal [n!case several prizes]
Probe [tdef] agrees with goal [n!case several prizes]
Probe [t] agrees with goal [n several prizes]
[c[c][t[n several prizes][t[t][v[v be][a[a likely][t[n several
prizes][t[tdef][v[v be][a[a ed][V[V award][n several
prizes]]]]]]]]]]
```

A Worked Example



A Worked Example



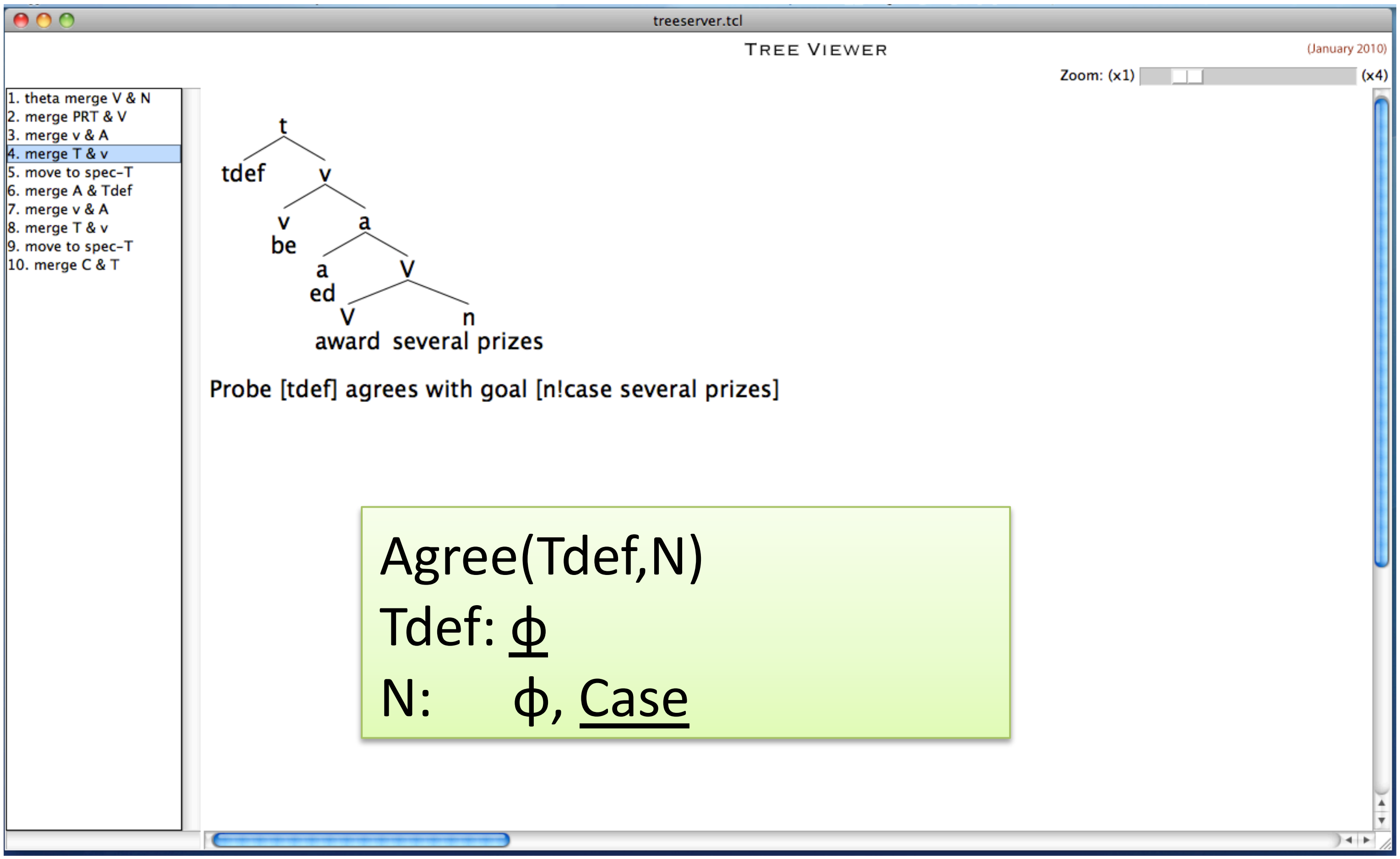
Notation: !case means feature is unvalued

Agree(a,N)

-ed: $\underline{\phi}$, $\underline{\text{Case}}$

N: ϕ, $\underline{\text{Case}}$

A Worked Example

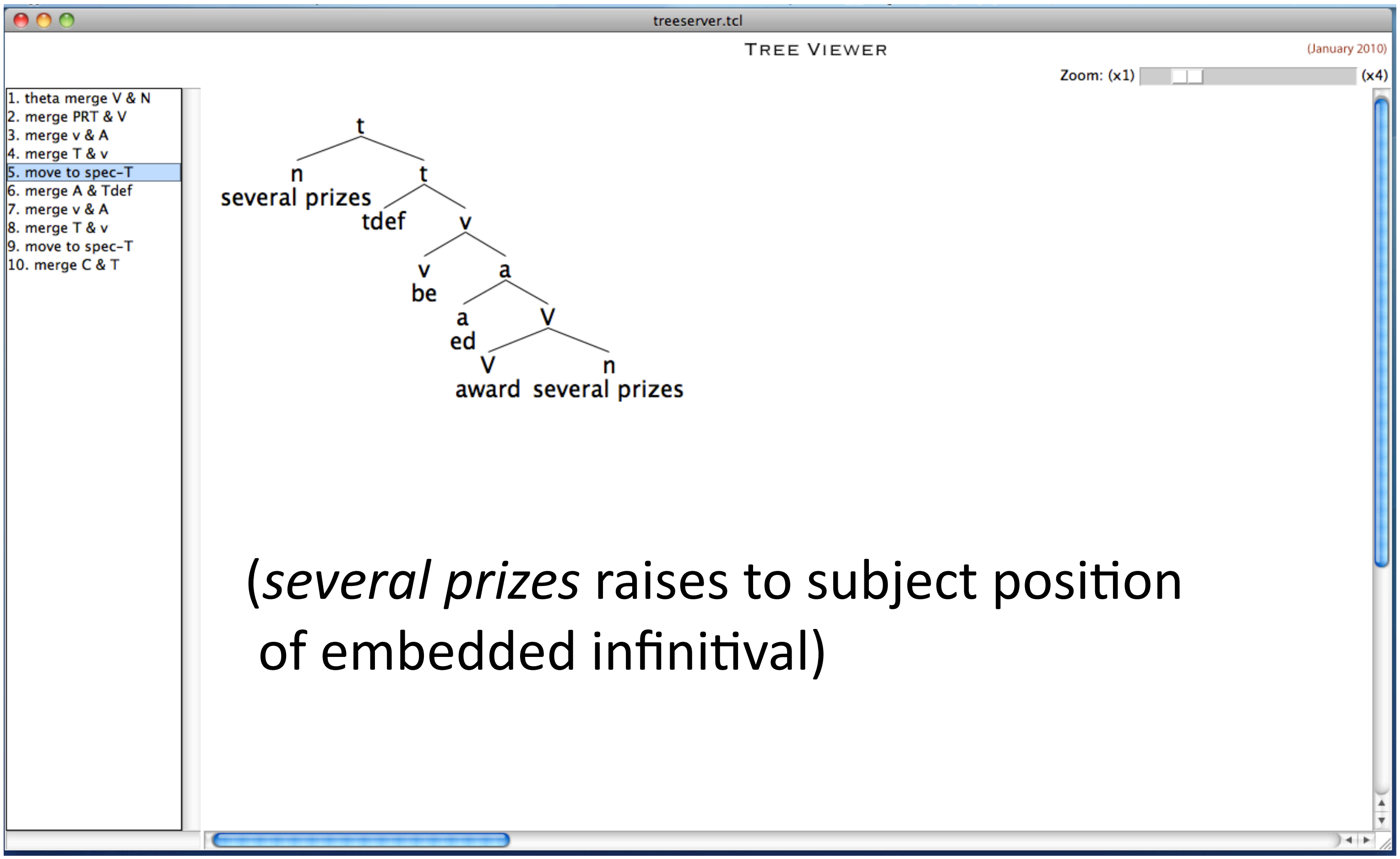


Agree(Tdef,N)

Tdef: <u>ϕ</u>

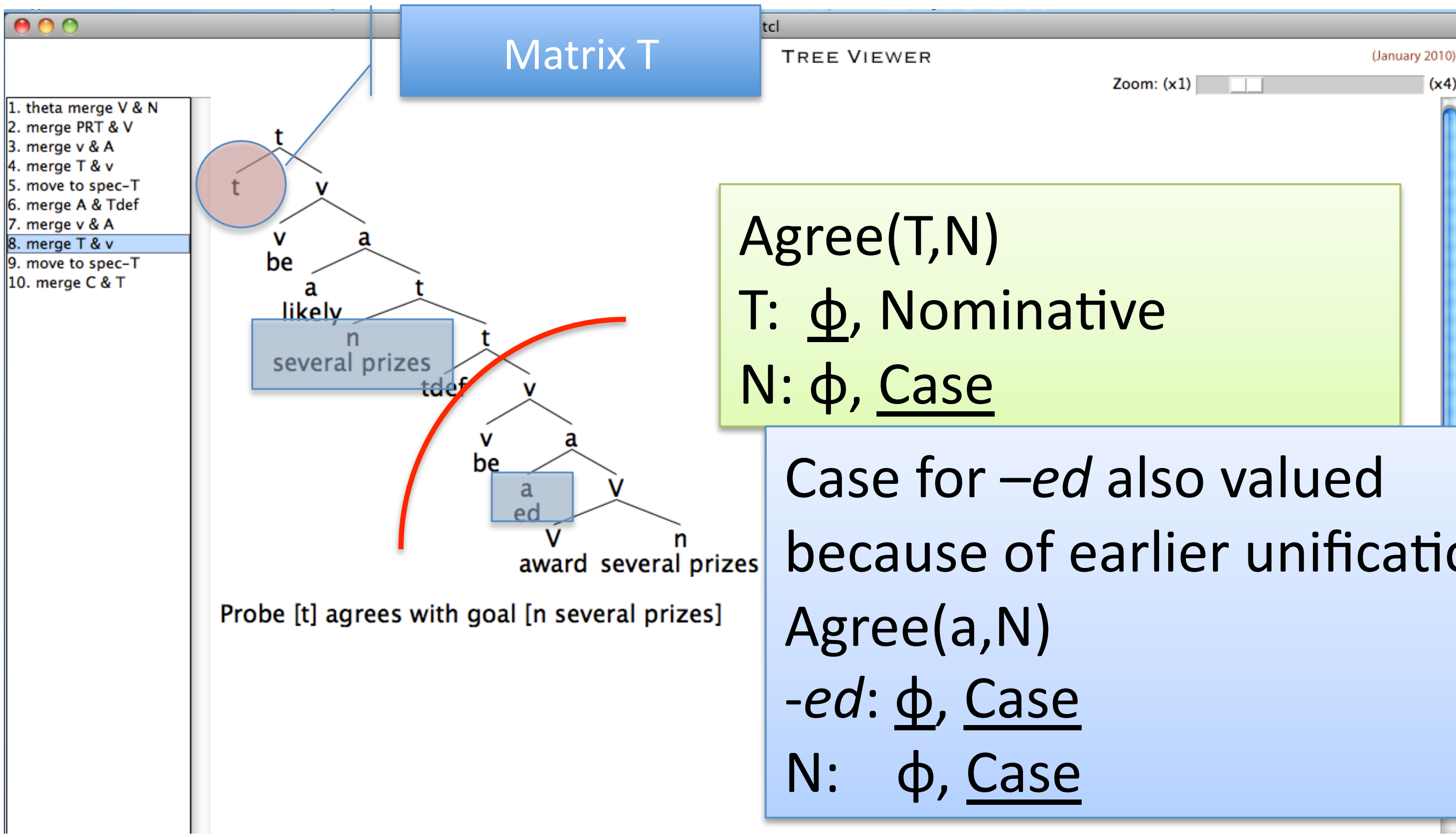
N: ϕ, <u>Case</u>

A Worked Example



(*several prizes* raises to subject position of embedded infinitival)

A Worked Example



Agree(T,N)
T: ϕ, Nominative
N: ϕ, Case

Case for *–ed* also valued
because of earlier unification step
Agree(a,N)
-ed: ϕ, Case
N: ϕ, Case

Unification presents an possible advantage: computation is more local

Probing with multiple goals

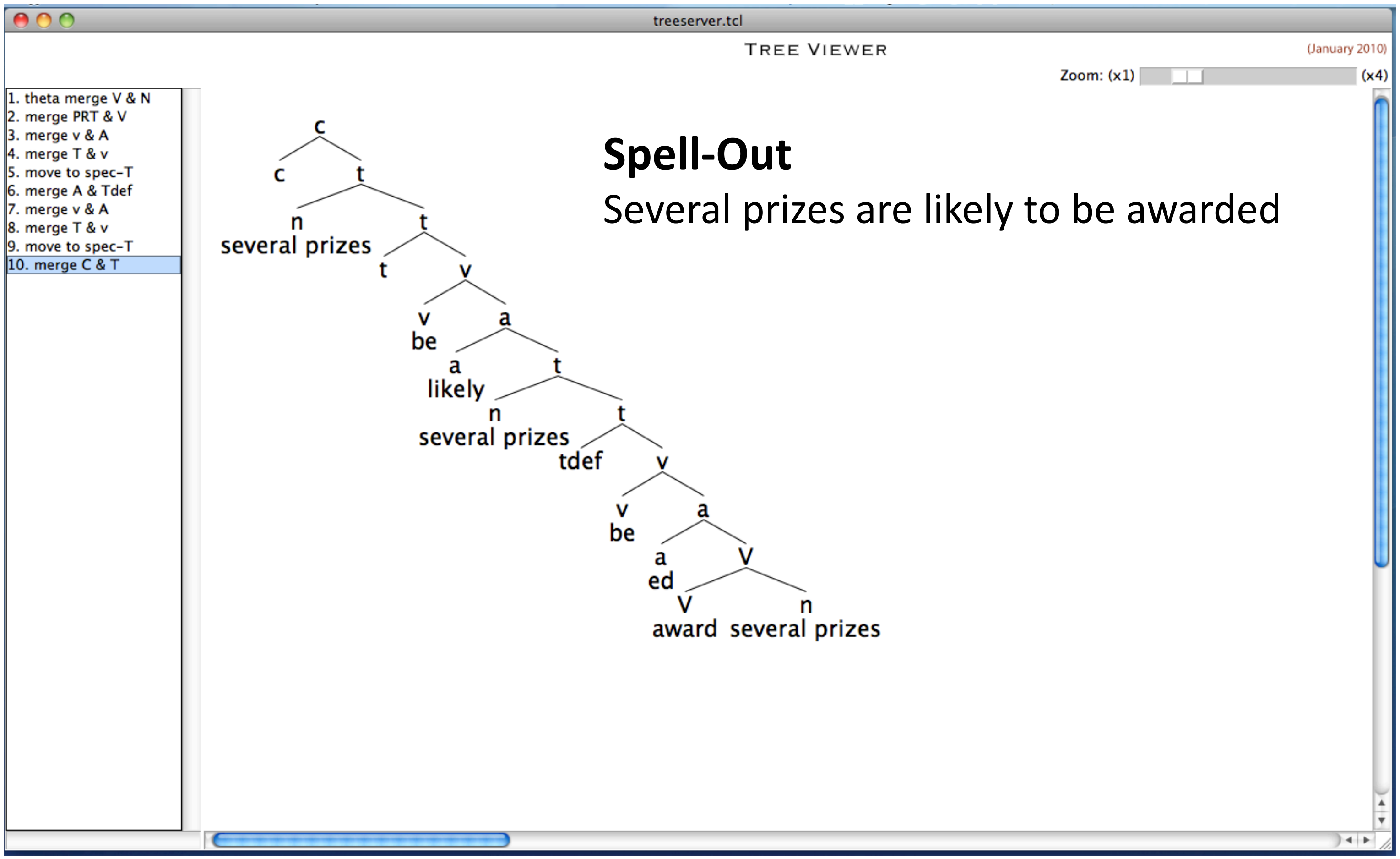
What about Prt? Its ϕ-features are deleted at stage α and should therefore be invisible to Match by the probe. Case of Prt cannot be valued and the derivation crashes, contrary to fact.

The problem is overcome if Spell-Out takes place at the strong-phase level. Then the ϕ-features of Prt are still visible at stage β of the cycle, though deleted; they disappear at the strong-phase level CP or *v*P, as the phase is transmitted to the phonological component. At stage α of the cycle, the ϕ-features of Prt are valued by Prt-DO matching, as just discussed. At the next stage, the probe T/*v* matches the (still visible) goal Prt, valuing its Case feature; and the probe matches the goal DO, valuing the Case feature of DO as well as its own features (since DO is ϕ-complete).

compare with…

(Chomsky 2001)

A Worked Example



Another Example

Consider also the derivation of

- There are likely to be awarded several prizes (= 4(b)(i))

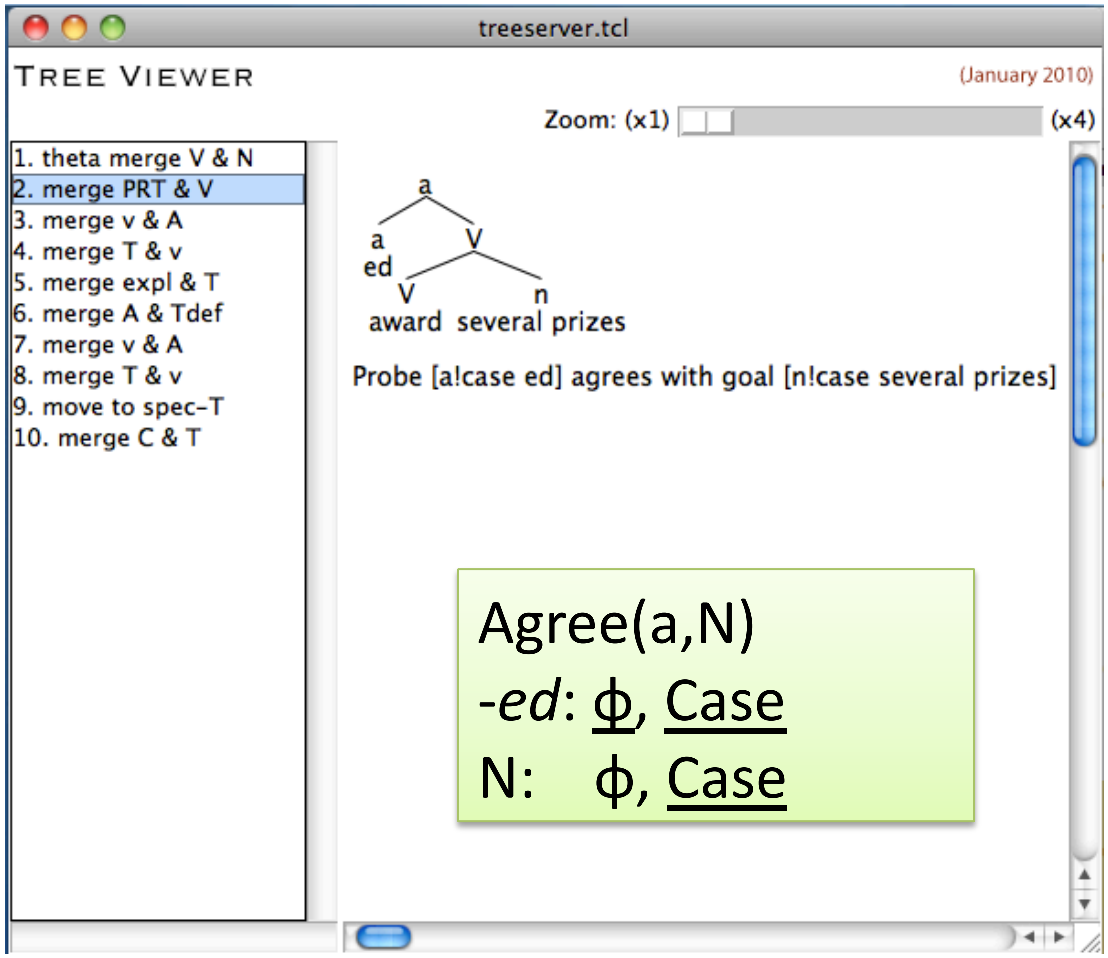
pleonastic ***there***: ϕ-incomplete
(Person only)

Agree(a,N)
-ed: <u>ϕ</u>, <u>Case</u>
N: ϕ, <u>Case</u>

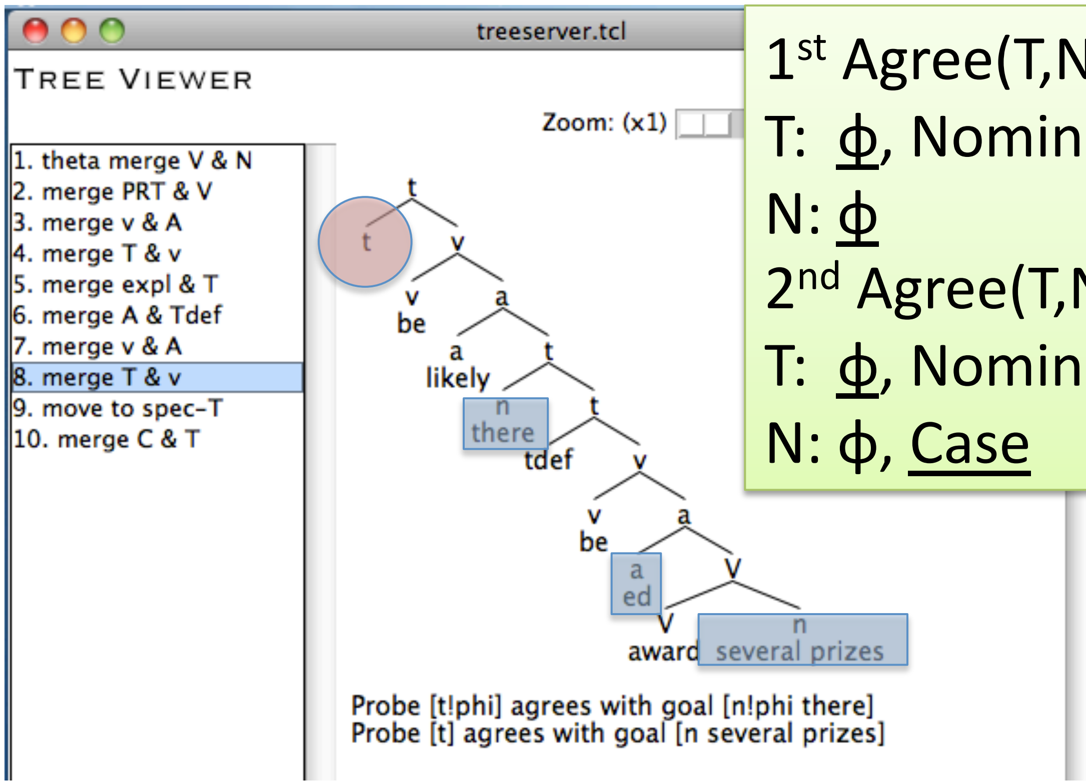
```
?- parse([be,likely,there,be,ed,award,several,prizes]).
Probe [a!case ed] agrees with goal [n!case several prizes]
Probe [tdef] agrees with goal [n!case several prizes]
Probe [t!phi] agrees with goal [n!phi there]
Probe [t] agrees with goal [n several prizes]
[c[c][t[n there][t[t][v[v be][a[a likely][t[n there][t[tdef][v[v
   be][a[a ed][V[V award][n several prizes]]]]]]]]]]]
```

Agree(T,N)
T: <u>ϕ</u>
N: <u>ϕ</u>

Another Example



Another Example



1st Agree(T,N)

T: ϕ, Nominative

N: ϕ

2nd Agree(T,N)

T: ϕ, Nominative

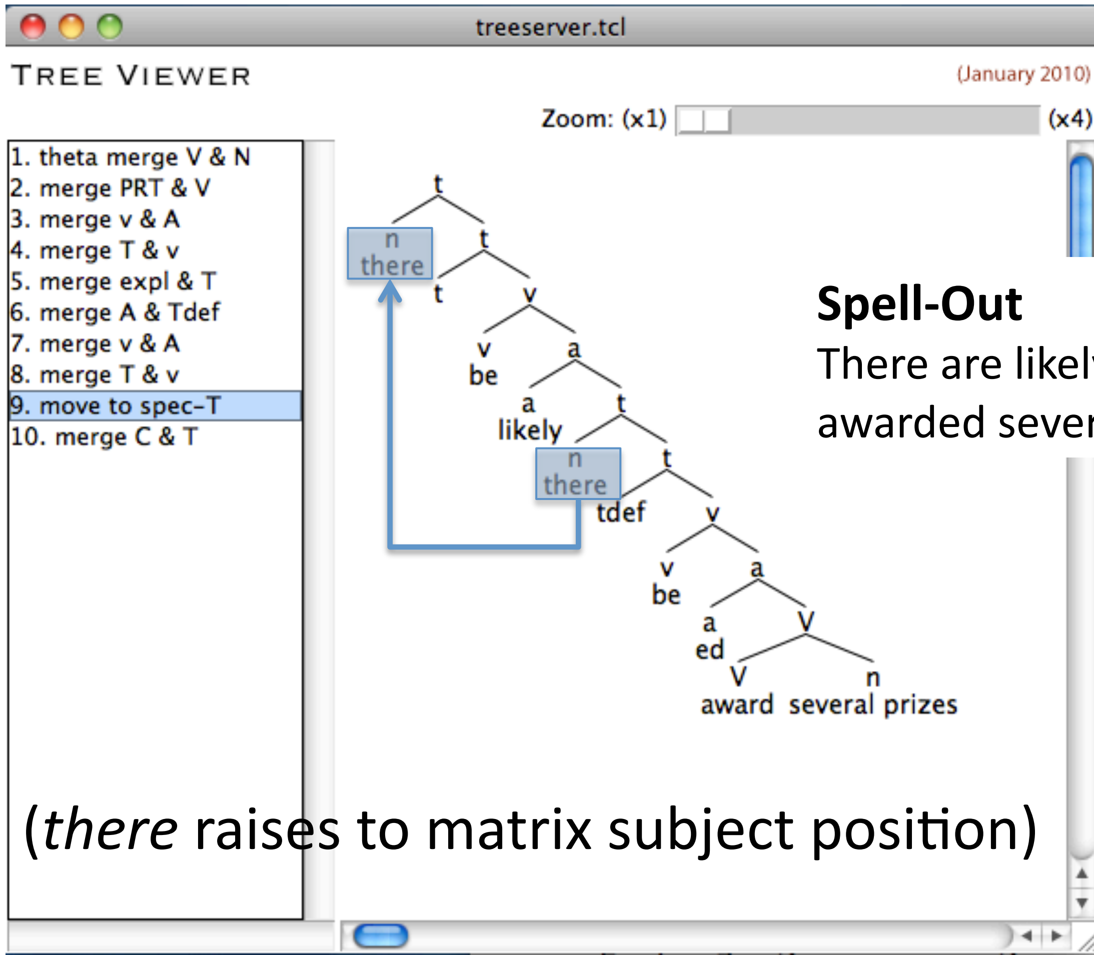
N: ϕ, Case

there : ϕ-feature indirectly valued by 2nd Agree(T,N)

-ed: Case also indirectly valued by 2nd Agree(T,N)

However, T must probe past *there* all the way on down

Another Example



Spell-Out
There are likely to be awarded several prizes

(*there* raises to matrix subject position)

Examples

assume, are raising constructions and their exceptional-Case-marking (ECM) counterparts, as shown schematically in (4a), where β is the matrix clause, α is an infinitival with YP a verbal phrase (the case most relevant here), and P is the probe: T with a raising verb (case (4b)), *v* with an ECM transitive verb (case (4c)).[10]

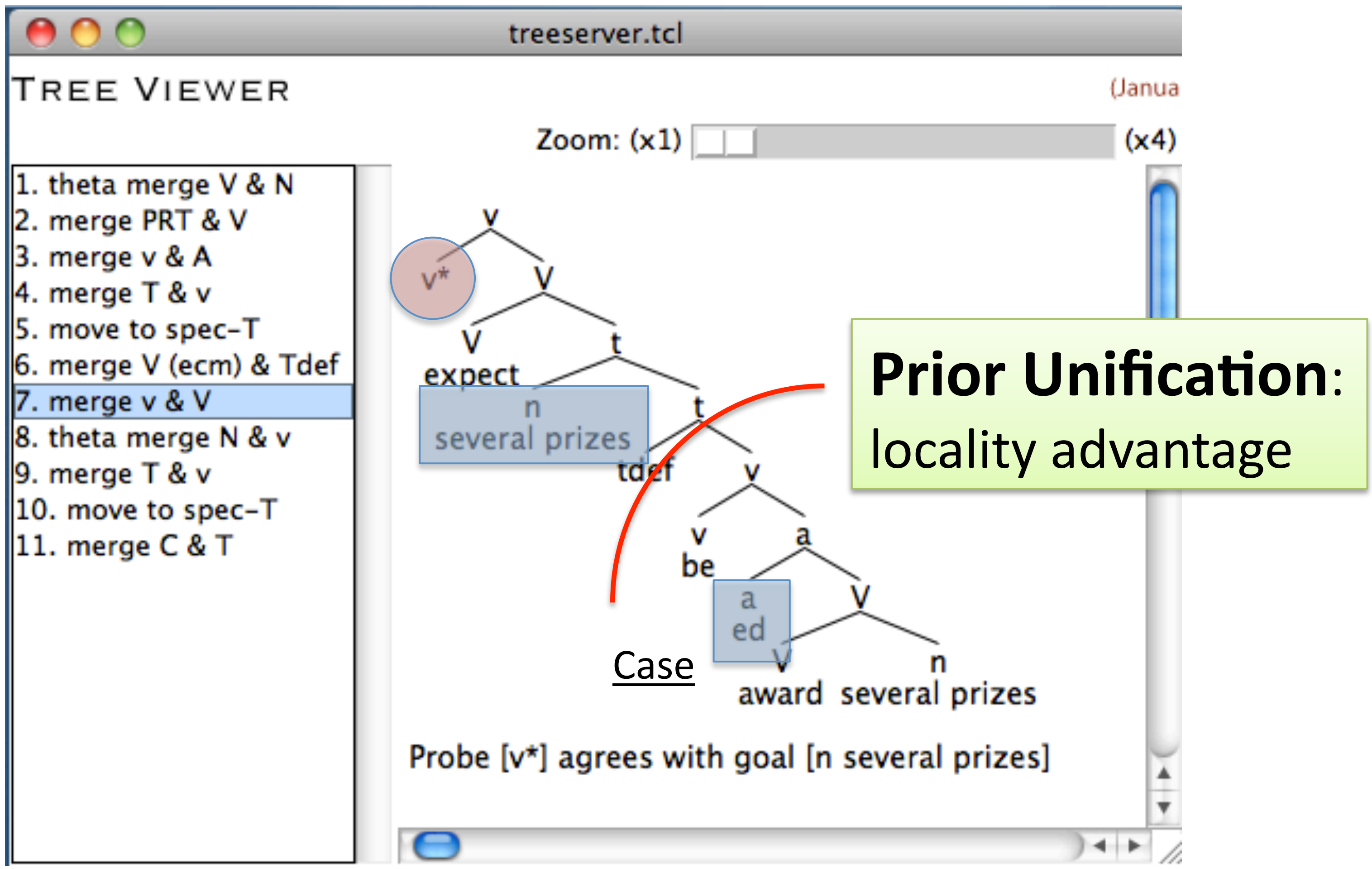
(4) a. $[_\beta$ P $[_\alpha$ [Subj [H YP]]]]
    b. i. there are likely to be awarded several prizes
        ii. several prizes are likely to be awarded
    c. i. we expect there to be awarded several prizes
        ii. we expect several prizes to be awarded

- Examples (ECM):
  - (i) we expect there to be awarded several prizes
  - (ii) we expect several prizes to be awarded

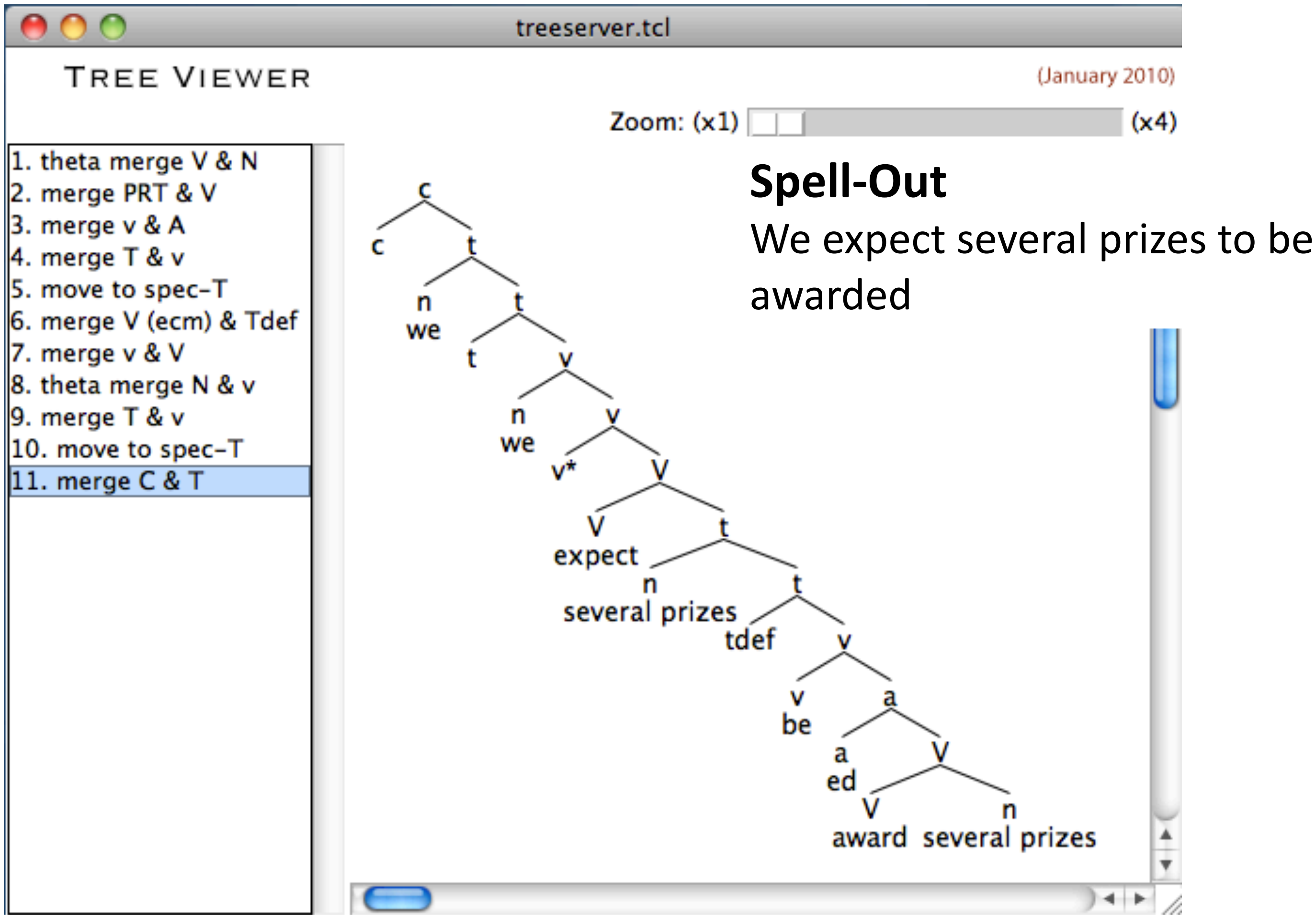
[Spec, T_{def}] in (4b) and in (4c) (see MI). In (4bi) and (4ci), the EPP-feature of T_{def} is satisfied by Merge of expletive; in (4bii) and (4cii), by raising of the direct object.

(Chomsky 2001)

Example: *we expect several prizes to be awarded*



Example: *we expect several prizes to be awarded*



Conclusions

- described a (non-trivial) implementation of a probe-goal system for Case and verbal inflection
- we want to (1) minimize # operations, and (2) localize goal search as far as possible
- unification for uninterpretable feature valuation can help both (1) and (2)