Modeling Linguistic Theory on a Computer: From GB to Minimalism

Sandiway Fong
Dept. of Linguistics
Dept. of Computer Science
THE UNIVERSITY OF ARIZONA
TUCSON ARIZONA

Outline

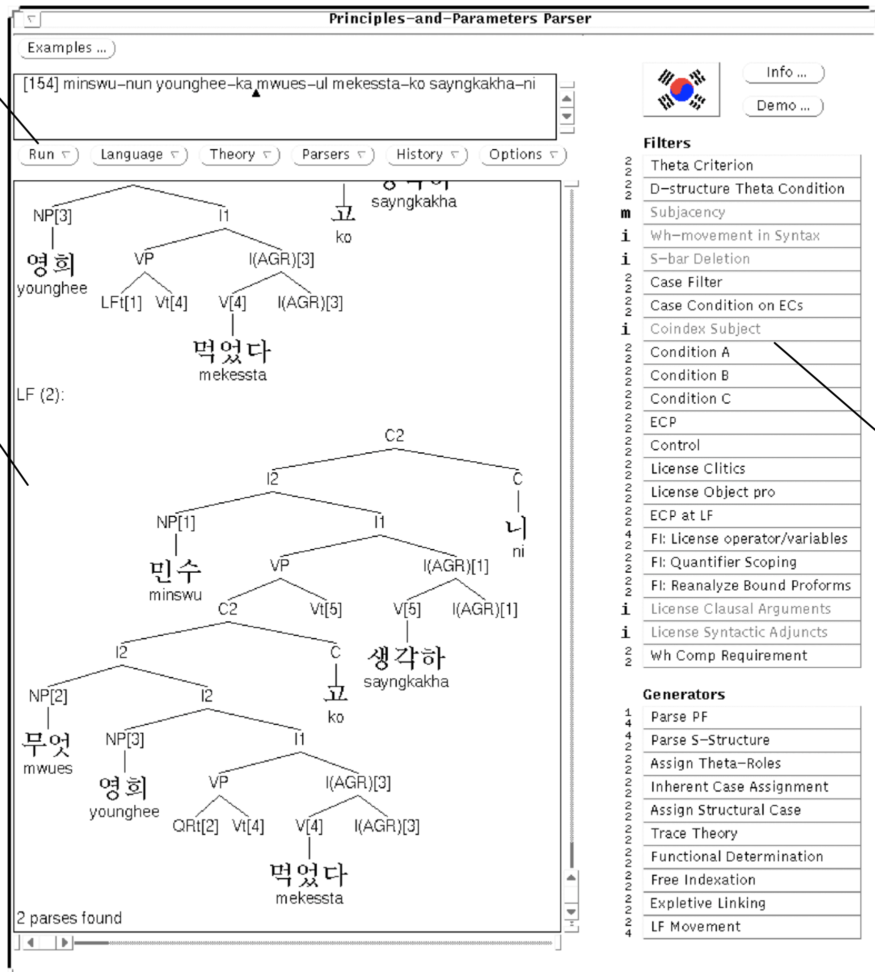
- **Mature system: PAPPI**
  - parser in the principles-and-parameters framework
  - principles are formalized and declaratively stated in Prolog (logic)
  - principles are mapped onto general computational mechanisms
  - recovers all possible parses
  - (free software, recently ported to MacOS X and Linux)
  - (see http://dingo.sbs.arizona.edu/~sandiway/)

- **Current work**
  - introduce a left-to-right parser based on the probe-goal model from the Minimalist Program (MP)
  - take a look at modeling some data from SOV languages
    - relativization in Turkish and Japanese
    - psycholinguistics (parsing preferences)
  - (software yet to be released...)

PAPPI: Overview

sentence

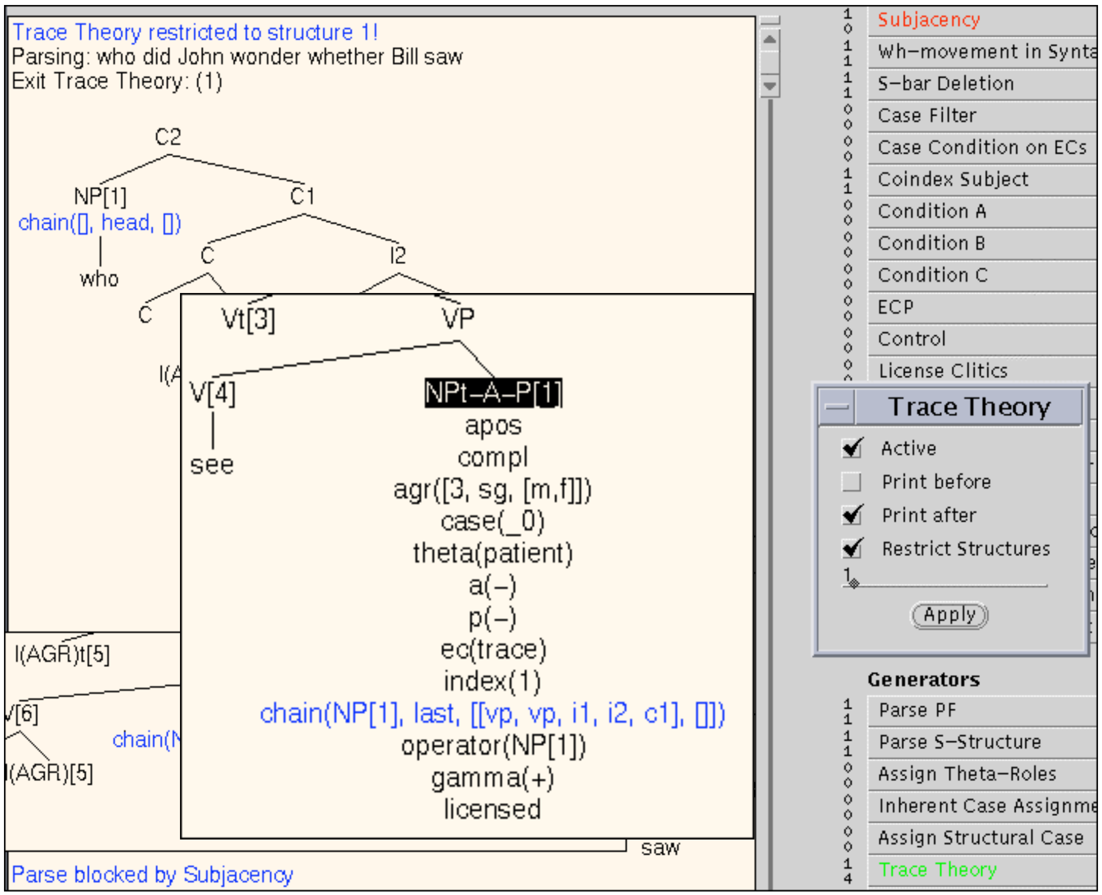
syntactic represent ations



- user's viewpoint

parser operations corresponding to linguistic principles (= theory)

PAPPI: Overview



- **parser operations can be**
  - turned on or off
  - metered
- **syntactic representations can be**
  - displayed
  - examined
    - in the context of a parser operation
  - dissected
    - features displayed

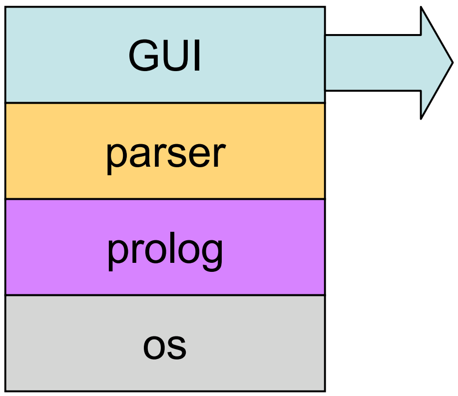
PAPPI: Coverage

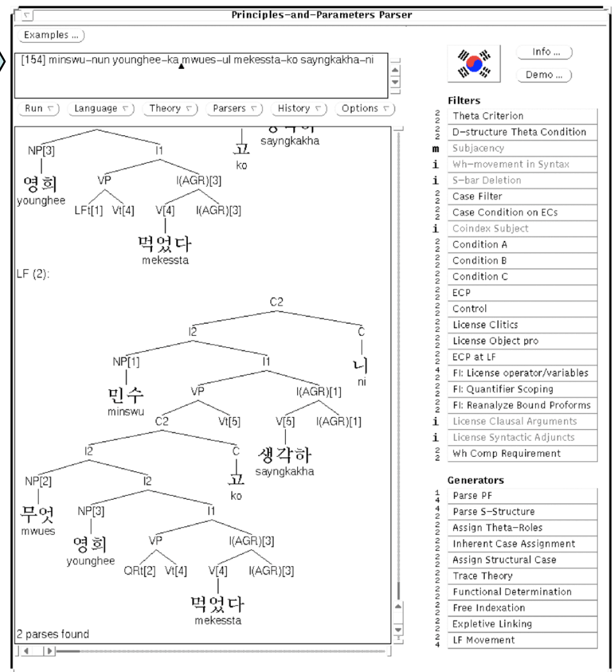
- **supplied with a basic set of principles**
  - X'-based phrase structure, Case, Binding, ECP, Theta, head movement, phrasal movement, LF movement, QR, operator-variable, WCO
  - handles a couple hundred English examples from Lasnik and Uriagereka's (1988) *A Course in GB Syntax*

- **more modules and principles can be added or borrowed**
  - VP-internal subjects, NPIs, double objects *Zero Syntax* (Pesetsky, 1995)

  - Japanese (some Korean): head-final, pro-drop, scrambling
  - Dutch (some German): V2, verb raising
  - French (some Spanish): verb movement, pronominal clitics
  - Turkish, Hungarian: complex morphology
  - Arabic: VSO, SVO word orders

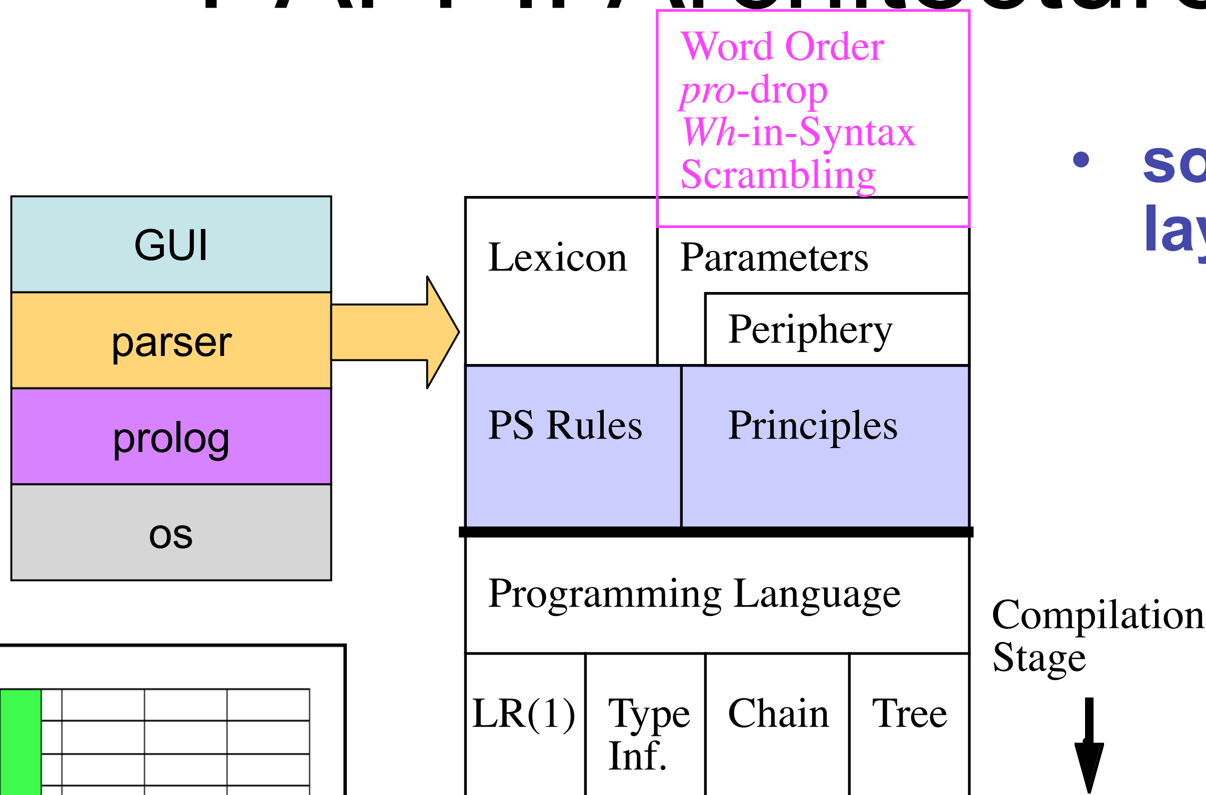
PAPPI: Architecture



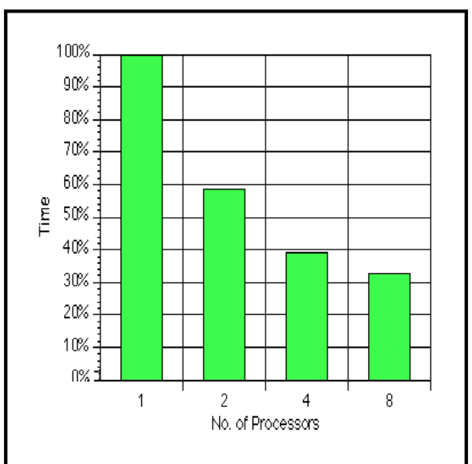


- **software layers**

PAPPI: Architecture

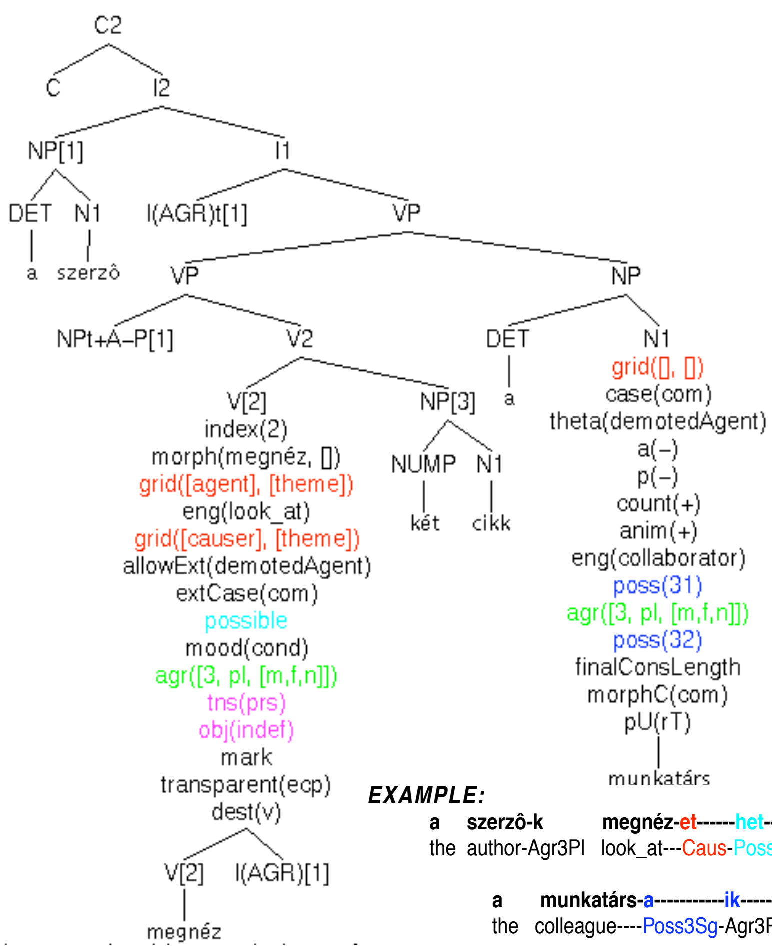


- **software layers**
  - competing parses can be run in parallel across multiple machines



PAPPI: Machinery

Parsing: a szerzôk megnézethetnének két cikket a munkatársaikkal
Exit Contraction: (1) a szerzô pl megnéz causative possibility cond agr3Pl két cikk acc a munkatárs poss3Sg pl poss3Pl lengthenedFinalConsonant com
Exit Parse PF: (1) [DET a][N szerzô][V megnéz]1 [NUM két][N cikk][DET a][N munkatárs] $
LF (1):



EXAMPLE:

a szerzô-k megnéz-et------het-----∅--------né-----nek----∅ két cikk---et
the author-Agr3Pl look_at---Caus-Possib-tns(prs)-Cond-Agr3Pl-Obj(indef) two article-Acc

a munkatárs-a----------ik--------------------kal
the colleague----Poss3Sg-Agr3Pl+Poss3Pl-LengdFC+Com

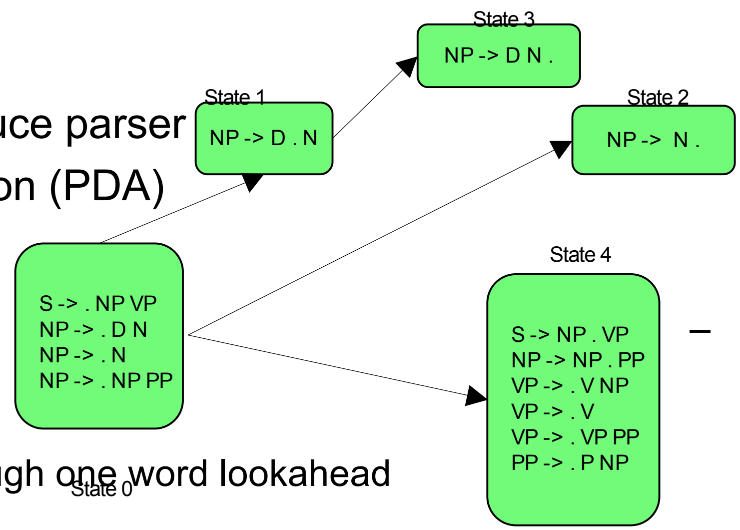
- **morphology**
  - simple morpheme concatenation
  - morphemes may project or be rendered as features
- (example from the Hungarian implementation)

PAPPI: LR Machinery

- **specification**
  - rule XP -> [XB|spec(XB)] ordered specFinal st max(XP), proj(XB,XP).
  - rule XB -> [X|compl(X)] ordered headInitial(X) st bar(XB), proj(X,XB), head(X).
  - rule v(V) moves_to i provided agr(strong), finite(V).
  - rule v(V) moves_to i provided agr(weak), V has_feature aux.

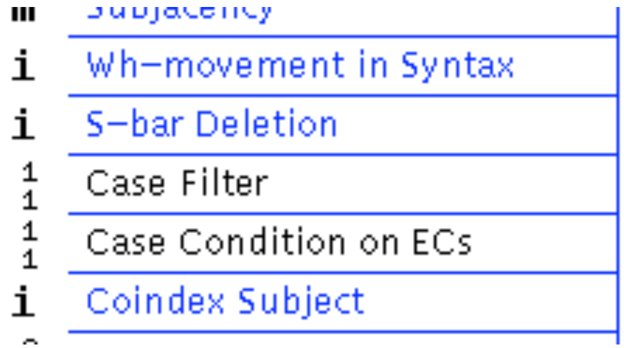
- **phrase structure**
  - parameterized X'-rules
  - head movement rules

- **implementation**
  - bottom-up, shift-reduce parser
  - push-down automaton (PDA)
  - stack-based merge
    - shift
    - reduce
  - canonical LR(1)
    - disambiguate through one word lookahead



- rules are not used directly during parsing for computational efficiency
  - mapped at compile-time onto LR machinery

PAPPI: Machine Parameters



- **specification**
  - coindexSubjAndINFL **in_all_configurations** CF **where** **specIP**(CF,Subject) **then** coindexSI(Subject,CF).
  - subjacency **in_all_configurations** CF **where** isTrace(CF), **upPath**(CF,Path) **then** lessThan2BoundingNodes(Path)

- **implementation**
  - use type inferencing defined over category labels
    - *figure out which LR reduce actions should place an outcall to a parser operation*
  - subjacency can be called during chain aggregation

- **selected parser operations may be integrated with phrase structure recovery or chain formation**
  - machine parameter
  - however, not always efficient to do so

PAPPI: Chain Formation

- **specification**
  - assignment of a chain feature to constituents

```
chain(NP[1], last, [[pp, vp, vp, i1], []])
chain(NP[1], medial, [[c2, vp, vp, i1, i2, c1], []])
chain([], head, [])
```

- **combinatorics**
  - exponential growth

NPs	Indexings	NPs	Indexings
1	1	7	877
2	2	8	4140
3	5	9	21147
4	15	10	115975
5	52	11	678570
6	203	12	4123597

Upper-bounded by *Bell's Exponential Number*

$$B_n = \sum_{m=1}^{n} \sum_{k=0}^{m} \frac{(-1)^{m-k}}{(m-k)!\,k!} k^n \qquad \frac{{m_n}^n e^{m_n - n - \frac{1}{2}}}{\sqrt{\ln n}}$$

$$m_n \ln m_n = n - \frac{1}{2}$$

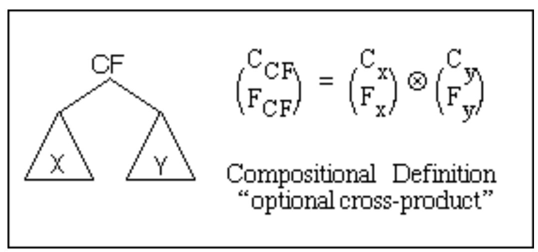
- **recovery of chains**
  - compute all possible combinations
    - each empty category optionally participates in a chain
    - each overt constituent optionally heads a chain

PAPPI: Chain Formation

- **specification**
  - assignment of a chain feature to constituents

```
chain(NP[1], last, [[pp, vp, vp, i1], []])
chain(NP[1], medial, [[c2, vp, vp, i1, i2, c1], []])
chain([], head, [])
```

- **implementation**
  - possible chains compositionally defined
  - incrementally computed
  - bottom-up



  - allows parser operation merge

- **recovery of chains**
  - compute all possible combinations
    - each empty category optionally participates in a chain
    - each overt constituent optionally heads a chain

PAPPI: Chain Formation

- **specification**
  - assignment of a chain feature to constituents

```
chain(NP[1], last, [[pp, vp, vp, i1], []])
chain(NP[1], medial, [[c2, vp, vp, i1, i2, c1], []])
chain([], head, [])
```

- **merge constraints on chain paths**

m	Subjacency
m	Lowering Filter
i	Wh–movement in Syntax

  - loweringFilter **in_all_configurations** CF **where** isTrace(CF), downPath(CF,Path) **then** Path=[].
  - subjacency **in_all_configurations** CF **where** isTrace(CF), **upPath**(CF,Path) **then** lessThan2BoundingNodes(Path)

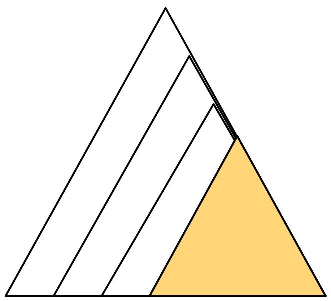
- **recovery of chains**
  - compute all possible combinations
    - each empty category optionally participates in a chain
    - each overt constituent optionally heads a chain

PAPPI: Domain Computation

- **specification**
  - gc(X) **smallest_configuration** CF **st** cat(CF,C), member(C,[np,i2])
  - **with_components**
  - X,
  - G **given_by** governs(G,X,CF),
  - S **given_by** accSubj(S,X,CF).

- **implementing**
  - Governing Category (GC):
  - GC(α) is the smallest NP or IP containing:
  - (A) α, and
  - (B) a governor of α, and
  - (C) an accessible SUBJECT for α.

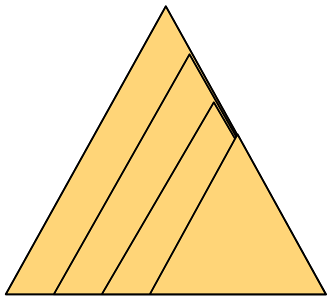
- **minimal domain**
  - incremental
  - bottom-up



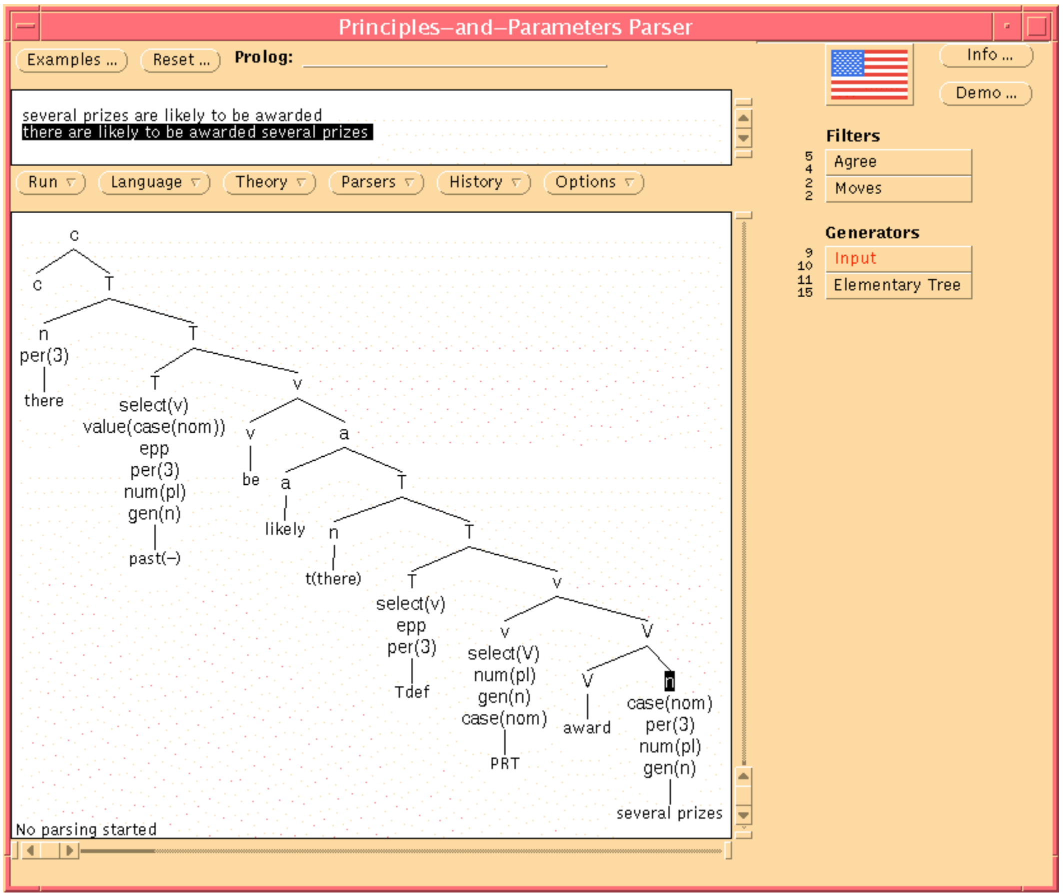
PAPPI: Domain Computation

- **specification**
  - gc(X) **smallest_configuration** CF **st** cat(CF,C), member(C,[np,i2])
  - **with_components**
  - X,
  - G **given_by** governs(G,X,CF),
  - S **given_by** accSubj(S,X,CF).
- **used in**
  - Binding Condition A
    - An anaphor must be A-bound in its GC
  - conditionA **in_all_configurations** CF **where**
  - anaphor(CF) **then** gc(CF,GC), aBound(CF,GC).
  - anaphor(NP) :- NP has_feature apos, NP has_feature a(+).

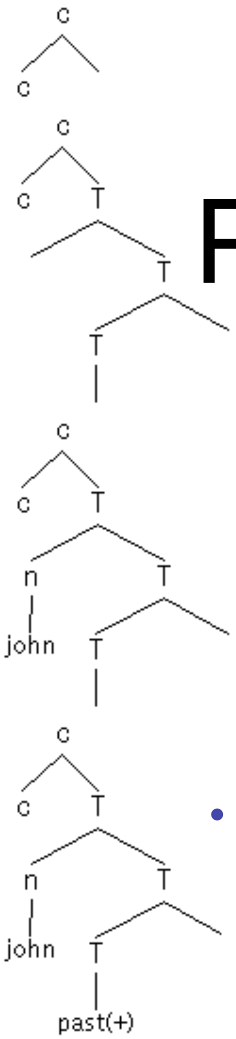
- **minimal domain**
  - incremental
  - bottom-up



Probe-Goal Parser: Overview



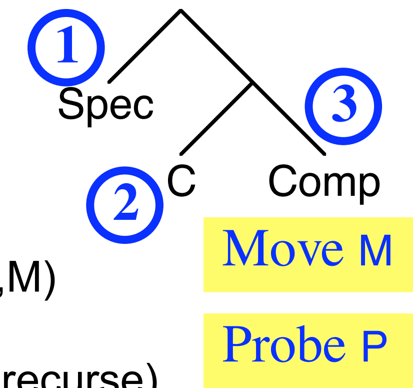
- **strictly incremental**
  - left-to-right
  - uses elementary tree (eT) composition
    - guided by selection
    - open positions filled from input
  - epp
  - no bottom-up merge/move
- **probe-goal agreement**
  - uninterpretable interpretable feature system



Probe-Goal Parser: Selection

- **recipe**

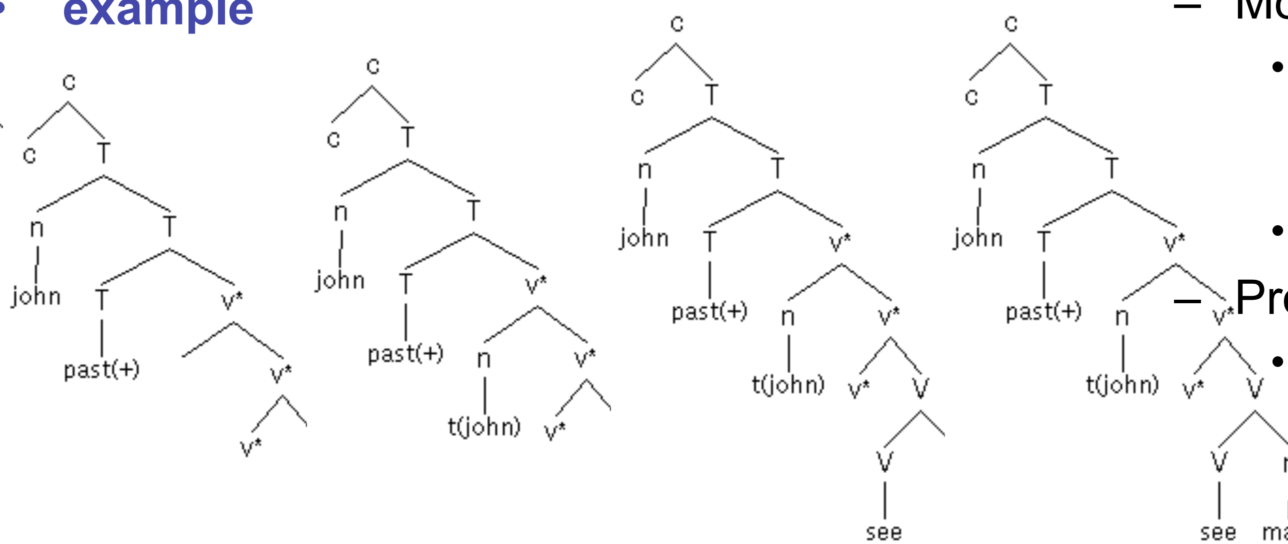
start(c)
pick eT headed by c
from input (or M)
  fill Spec, run agree(P,M)
  fill Head, update P
  fill Comp (c select c', recurse)



- **select drives derivation**
  - left-to-right

- **memory elements**
  - MoveBox (M)
    - emptied in accordance with theta theory
    - filled from input
  - ProbeBox (P)
    - current probe

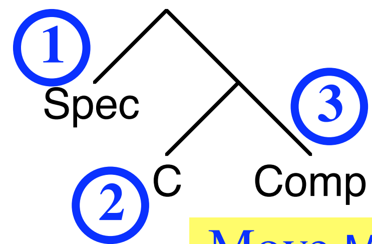
- **example**



Probe-Goal Parser: Selection

- **recipe**

start(c)
pick eT headed by c
from input (or M)
  fill Spec, run agree(P,M)
  fill Head, update P
  fill Comp (c select c', recurse)



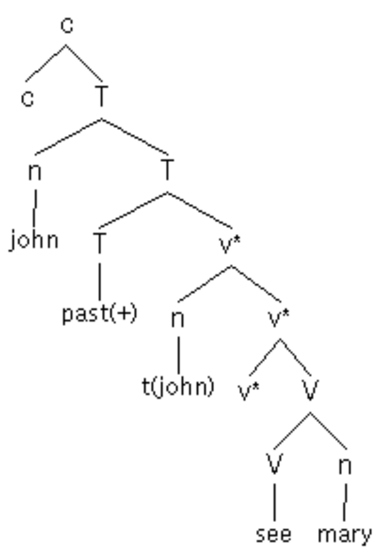
Move M

Probe P

- **select drives derivation**
  – left-to-right

- **memory elements**
  – MoveBox (M)
    - emptied in accordance with theta theory
    - filled from input
  – ProbeBox (P)
    - current probe

- **example**



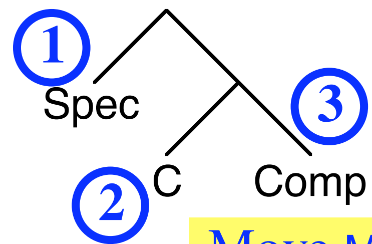

```
agree
```
ϕ-features → probe
case → goal

- **note**
  – extends derivation to the right
    - similar to Phillips (1995)

Probe-Goal Parser: Selection

- **recipe**

start(c)
pick eT headed by c
from input (or M)
  fill Spec, run agree(P,M)
  fill Head, update P
  fill Comp (c select c', recurse)



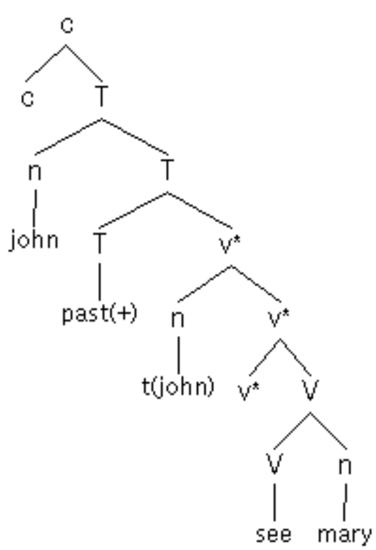
Move M

Probe P

- **select drives derivation**
  - left-to-right

- **memory elements**
  - MoveBox (M)
    - emptied in accordance with theta theory
    - filled from input
  - ProbeBox (P)
    - current probe

- **example**



- **note**
  - no merge/move
    - cf. Minimalist Grammar. Stabler (1997)

```
agree
```
ϕ-features → probe
case → goal

Probe-Goal Parser: Lexicon

lexical item	**properties**	**uninterpretable features**	**interpretable features**
v* (transitive)	select(V) spec(select(N)) value(case(acc))	per(P) (epp) num(N) gen(G)	
v (unaccusative)	select(V)		
v# (unergative)	select(V) spec(select(N))		
PRT. (participle)	select(V)	num(N) case(C) gen(G)	
V (trans/unacc)	select(N)		
V (unergative)			
V (raising/ecm)	select(T(def))		

Probe-Goal Parser: Lexicon

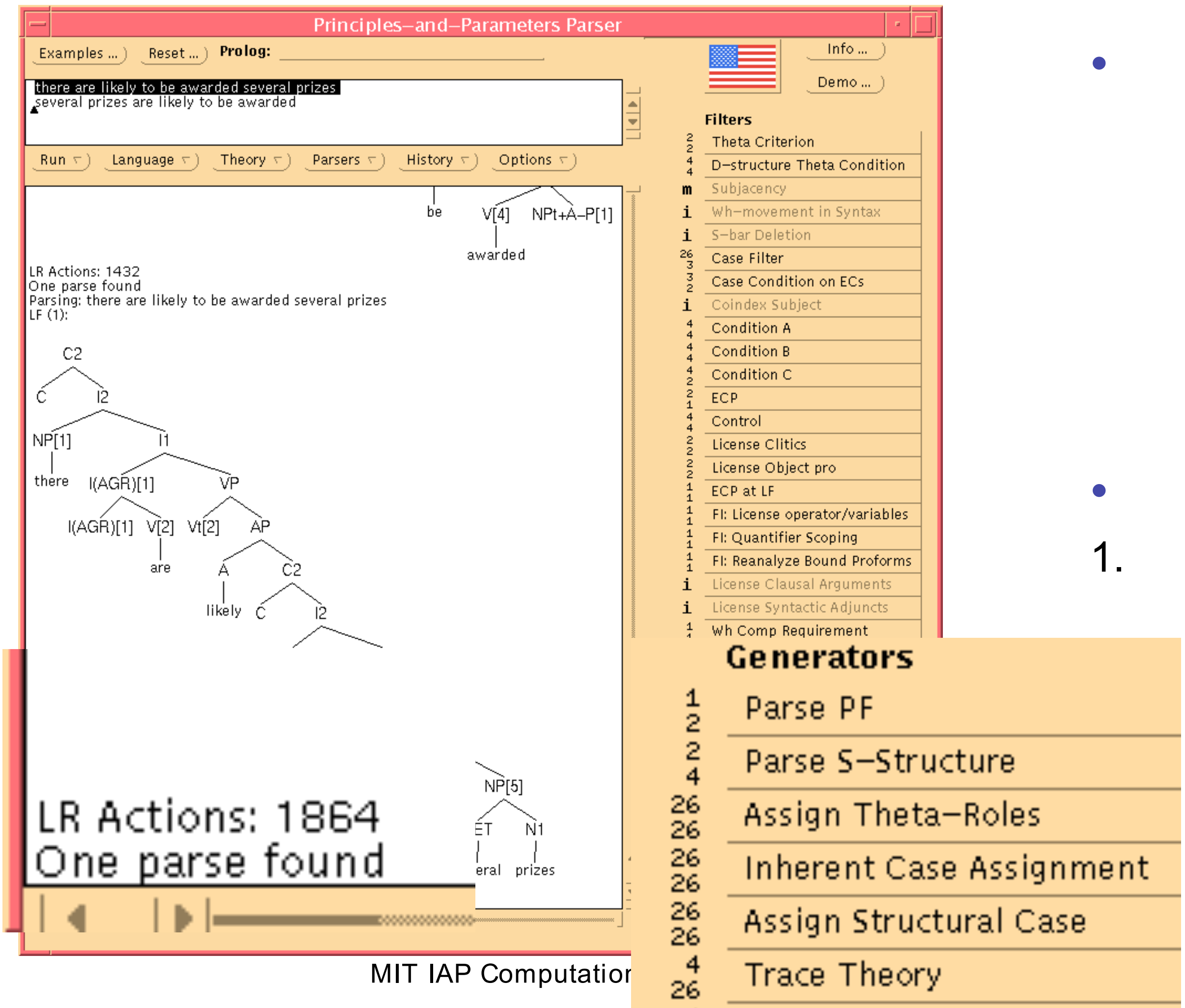
lexical item	**properties**	**uninterpretable features**	**interpretable features**
T	select(v) value(case(nom))	per(P) epp num(N) gen(G)	
T(def) (ϕ-incomplete)	select(v)	per(P) epp	
c	select(T)		
c(wh)	select(T)	q epp	wh
N (referential)	select(N)	case(C)	per(P) num(N) gen(G)
N (wh)		case(C) wh	per(P) q num(N) gen(G)
N (expl)	select(T(def))	per(P)	

Probe-Goal Parser: Memory

- **MoveBox M Management Rules** Move M
  - (*implements theta theory*)
  1. Initial condition: *empty*
  2. Fill condition: *copy from input*
  3. Use condition: ***prefer** M over input*
  4. Empty condition: *M emptied when used at selected positions. EXPL emptied optionally at non-selected positions.*

- **examples** from *Derivation by Phase*. Chomsky (1999)
  1. several prizes are likely to be awarded
     - [c [c] [T *several prizes* [T [T past(-)] [v [v be] [a [a likely] [T c(*prizes*) [T [T] [v [v PRT] [V [V award] c(*prizes*)]]]]]]]]]
  2. there are likely to be awarded several prizes
     - [c [c] [T *there* [T [T past(-)] [v [v be] [a [a likely] [T c(*there*) [T [T] [v [v prt] [V [V award] *several prizes*]]]]]]]]]

Probe-Goal Parser vs. PAPPI



- **instrument parser operations**

- **examples**

1. several prizes are likely to be awarded

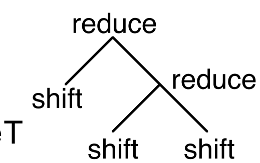
there are likely to be awarded several prizes

Probe-Goal Parser vs. PAPPI

example	structure building	agree/move vs. move-α
1.	15 eT/10 words	5/2
1. PAPPI	1864 LR ≈ 373 eT	26
2.	20 eT/16 words	7/7
2. PAPPI	1432 LR ≈ 286 eT	67

- **instrument parser operations**

exchange rate
5 LR ≡ 1 eT



examples

. several prizes are likely to be awarded

2. there are likely to be awarded several prizes

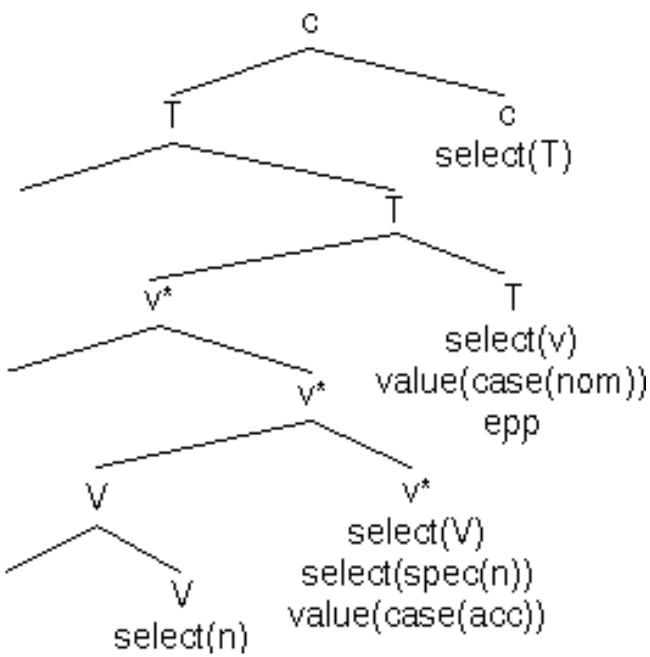
Probe-Goal Parser:
efficiency and preferences

- **MoveBox M Management Rule**
  3. Use condition: ***prefer*** *M over input*

- *How to expand the left-to-right model to deal with SOV languages and parsing preferences*?
  - look at some relativization data from Turkish and Japanese

- **efficiency**
  - choice point management
  - eliminate choice points

Probe-Goal Parser: SOV



- **assumptions**
  - posit simplex sentence structure
  - initially selection-driven
  - fill in open positions on left edge
    - left to right
  - possible continuations
  - 1: S O V *simplex sentence*
  - 2: [S O V]-REL ***V*** *complement clause*
  - 3: [S O V] ⇒ ***N*** *prenominal relative clause*

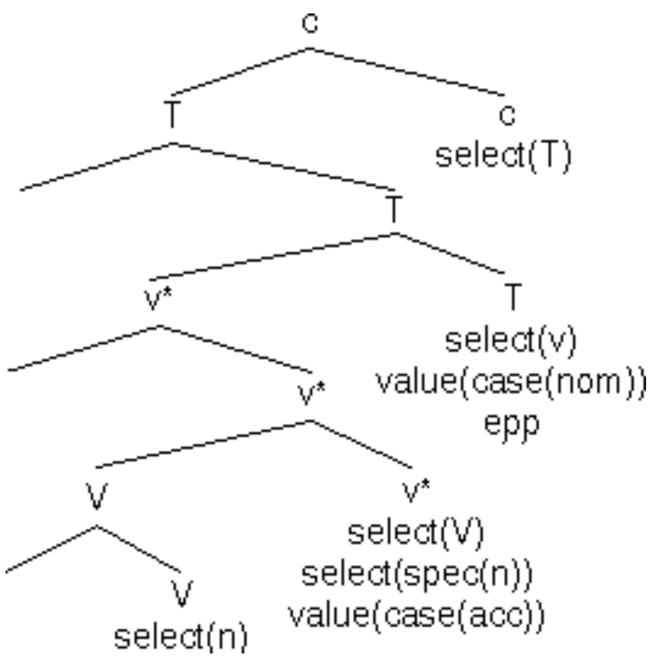
note

–don't posit unnecessary structure

–relative clauses are initially processed as main clauses with dropped arguments

–1 < 2 < 3, e.g. 2 < 3 for Japanese (Miyamoto 2002) (Yamashita 1995)

Probe-Goal Parser: SOV



- **assumptions**
  - posit simplex sentence structure
  - initially selection-driven
  - fill in open positions on left edge
    - left to right
  - possible continuations
  - 1: S O V *simplex sentence*
  - 2: [S O V]-REL ***V*** *complement clause*
  - 3: [S O V] ⇒ ***N*** *prenominal relative clause*

note

–**lack of expectation**

- [[[Op[[$_T$ S [$_v$ c(S) [$_V$ O V] v] T] c]]⇒S [_ [_ V]v]T]c]
- *in addition to the top-down (predictive) component*
- needs to be a bottom-up component to the parser as well

Probe-Goal Parser:
relative clauses

- **prenominal relative clause structure**
  - Turkish
    - [S-GEN O V-OREL-AGR] H
    - [S O-ACC V-SREL] H
    - OREL = -dUk
    - SREL = -An
  - Japanese
    - [S-NOM O V] H
    - [S O-ACC V] H
    - *no overt relativizer*

- **relativization preferences**
  - Turkish
    - *ambiguous Bare NP* (*BNP*)
    - BNP: BNP is object
    - BNP with possessive AGR: BNP is subject
  - Japanese
    - subject relative clauses easier to process
    - **scrambled object preference for relativization out of possessive object**

Ambiguity in Relativization (Turkish)
bare NPs and SREL

- **schema**
  - BNP V-SREL H
- **notes**
  - BNP = bare NP (not marked with ACC, same as NOM)
    - (1) indefinite object NP, i.e. [O [e **BNP** V-SREL]] ⇒H
    - (2) subject NP, i.e. [O [**BNP** e V-SREL]] ⇒H

general preference (subject relativization)

- ***e*** BNP V-SREL **H**

•**however** …

–Object relativization preferred, i.e. BNP ***e*** V-SREL **H** when BNP V together form a unit concept, as in:

•*bee sting*, *lightning strike* (pseudo agent incorporation)

Ambiguity in Relativization (Turkish)

possessor relativization and bare NPs

- **schema**
  - BNP-AGR V-SREL H (AGR indicates possessive agreement)
- **example** (Iskender, p.c.)
  - daughter-AGR see-SREL man
    the man whose daughter saw s.t./s.o.

general preference (*BNP as subject*)

- [*e* BNP]-AGR pro V-SREL **H**

- **notes**
  - BNP with AGR in subject position vs. in object position without
  - Object pro normally disfavored viz-a-viz subject pro
  - See also (Güngördü & Engdahl, 1998) for a HPSG account

Possessor Relativization (Japanese)

subject/object asymmetry

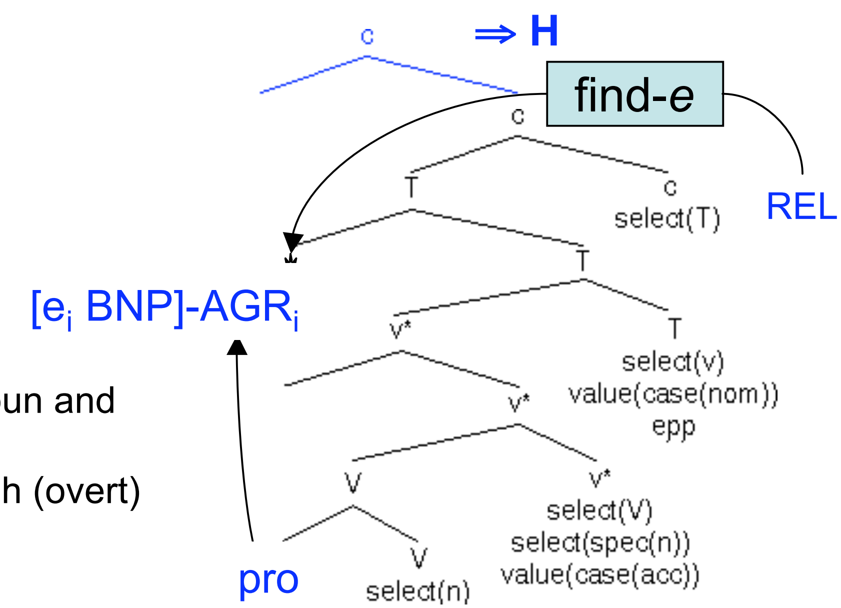
- **examples** (Hirose, p.c.)
- *also Korean* (K. Shin; S. Kang, p.c.)
  - subject
    - musume-ga watashi-o mita otoko
    - [**e** daughter]-NOM I-ACC see-PAST **man**

      the man whose daughter saw me
  - object
    - musume-o watashi-ga mita otoko
    - [**e** daughter]-ACC I-NOM *e* see-PAST **man**
    - *?*I-NOM [**e** daughter]-ACC see-PAST **man**

- **summary**
  - scrambled version preferred for object relativization case
    - *non-scrambled version is more marked*
  - in object scrambling, object raises to spec-T (Miyagawa, 2004)
  - possible difference wrt. inalienable/alienable possession in Korean

Probe-Goal Parser: A Model

- **initial expectation**
  - simple clause
  - top-down prediction
  - fill in left edge
  - insert *pro* as necessary
- **surprise**
  - triggers REL insertion at head noun and bottom-up structure
  - REL in Japanese (covert), Turkish (overt)
  - S O V (REL) **H**



- **functions of REL**
  - –introduces empty operator
  - –looks for associated gap (find-*e*) in predicted structure

doesn't work for Chinese:

object relativization preference (Hsiao & Gibson)