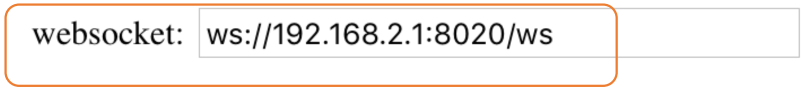
LING 696G: Lecture 4

Sandiway Fong

Free Merge Machine (FMM)

- Five copies of the application are running on a Macbook Pro in my office.
- Connect to Wifi network Free Merge Machine
- Machine has LAN address 192.168.2.1
- You can use one of the ports: 8025, 8024, 8023, 8022
- Open file `index_ua.html` and pick a port number for the websocket

FreeMergeMachine



Free Merge Machine (FMM)

```
<script>
var port = 8020;                    // NEW!
var websocket;
var steps = 1;                      // expansion limit
var c;                              // canvas
var i;                              // image
var ctx;                            // canvas context
```

```
<body>
  <div style="position:fixed; top:2px; right:2px; background-color:#FFF; width:300px" id="fixed">
    <label for='host' style="font-size:smaller">websocket:</label>
    <script>
      var host = "192.168.2.1"
      document.write("<input type='text' id='host' value=\"ws://" +
  host + ":" + port +  "/ws\" required  pattern='ws://.+?:\d+/ws' size=32>")
    </script>
  </div>
```

Modify `index_ua.html` for permanent change

Cheat Sheet

Key: expand

Notation:

Set Merge (SM): {α,β}; Pair Merge (IM): <α,β>, α is adjunct. α!F means α has unvalued feature F.
Workspace (WS) = Syntactic Object (SO) + unprocessed Lexical Items (LIs).
Initial WS: LIs = a list of heads ([...]) to be processed in order. 1st(LIs) denotes the first element. Initial SO = 1st(LIs).
Sub-WS: a sub-list defines a sub-WS. Compute a sub-SO that substitutes for the sub-list in the higher WS.

Derivation Tree (DT) examples:

Each line encodes one step of the DT. Formats: (1) `op SO Input` (non-leaf step) (2) `op *Reason SO Input` (blocked) (3) `op ✓ SO []` (convergent step)	Explanation: op = previous operation resulting in SO; Input = LIs remaining Reason = a restriction (or end) blocking SO (end: labeled SO computed but unvalued features still present)
esm SO: {{friend,n!case},like}, Input: [v*] 1▶ism SO: {{friend,n!case},{{friend,n!case},like}}, Input: [v*] 2▶esm SO: {{{friend,n},like},v*}, Input: [] (Click on ▶ to extend the derivation one step.)	Explanation: from SO {{friend,n!case},like}, there are two possible ways to proceed: (1) ism of {friend,n!case} (object shift), or (2) esm of v*.
SO: book, Input: [n!case,[d,the]] 1 epm *pmR SO: <n!case,book>, Input: [[d,the]] 2 epm *mergeR(ext) SO: <book,n!case>, Input: [[d,the]] 3 esm SO: {book,n!case}, Input: [[d,the]]	Explanation: (greyed out = blocked derivation) from SO book, epm of n!case is blocked (*) in 1 and 2 by restrictions pmR and mergeR, resp.; however, esm of n!case, option 3, is permitted.

Operations:	Operation	Restriction	Restriction	Restriction
esm: External SM {SO,1st(LIs)}	epm: External PM <SO,1st(LIs)> or <1st(LIs),SO>	pmR: *<α[uF],β> no unvalued features (uF) within adjunct α	dup: duplicate SOs eliminated	xmit: transmit INFL failure phase head, e.g. C or v*, transmits inflectional (INFL) features to lower head X, triggering Agree(X, β), β a goal
ism: Internal SM {α,SO}, α a sub-SO of SO	dws: Down WS begin computing sub-list	ipmR: disallow <{x,y},x> from {x,y}	loop: IM repetitions disallowed e.g. *ism α ism α, or *ism α ism β ism α ism β	cii: CI interface crash uninterpretable formulae: *<nP,nP>, *<dP,dP> (cf. <dP,nP>)
ipm: Internal PM (IPM) <SO,α>, α a sub-SO of SO	uws: Up WS end sub-list computation; SO to higher WS	mergeR: apply lexical restrictions e.g. a categorizer must be the 1st SM'ed head above a Root	unlabeled (SO): labeling algorithm: non-weak head X labels{X,YP}, R (root) and T weak weak head W labels {W,YP} if strengthened; X labels {X,R}; <ϕ,ϕ> labels {XP,{Y,ZP}} assuming identical ϕ-features for XP and Y, strengthened Y labels {Y,ZP} XP labels {XY,YP} if YP moves Stipulation: n* strengthens R in {n*,{R,XP}}	

grammar.pl

```
%% lexicon(name,features,functions)
%% functions:
%% mergeR(restrictions)
%%   mergeR(root)
%% value(feature)
%% phase(Type,INFLfeatures)  transmit INFLfeatures (Type)
```

```
%% roots
%% lexical roots must be categorized
lexicon(R,[infl(invis(_))],[aB(mergeR(cat))]) :- root(R).

%% function value theta role (unused)
%% roots must be categorized and agree is triggered by receiving INFL
lexicon(R,[infl(invis(_))],[aB(mergeR(cat))]) :- rootHasObject(R).

lexicon(a,[specific(-),num(1)],[aB(mergeR(cat_d))]).
lexicon(the,[specific(+)],[aB(mergeR(cat_d))]).
lexicon(two,[num(2)],[]).
```

grammar.pl

```
%% for labeling
weak(tpast).
weak(tpres).
weak(t).
weak('n*').

rootHasObject(like).
rootHasObject(friend).

root(john).
root(book).
root(me).
root(he).
root(she).
```

```
% Primitive types
type(t).
type(d).
type(v).
type(n).
type(c).
type(p).
type(a).
type(root).

type(n3sg,n). type(n3pl,n).
type(a,dRoot). type(the,dRoot). type(two,dRoot).
type(dRoot,root).
type('d*',d).
type('v*',v).
type('n*',n).
type(tpast,t).  type(tpres,t).
type(X,root) :- root(X).
type(X,root) :- rootHasObject(X).
```

grammar.pl

```
%% functional categories
%% mergeR(root) (categorize) can be done only once
%% aB(F) = F abbreviation, look up in abbreviates(F,Full)
lexicon(n,[case(_),phi([3,sg],K)],[aB(mergeR(root,_))]) :- key(K).
lexicon(n,[case(_),phi([3,pl],K)],[aB(mergeR(root,_))]) :- key(K).
lexicon(v,[],[aB(mergeR(root,_))]).
lexicon('v*',[],
        [phase(infl_acc,_),aB(mergeR(root,_))]).
lexicon('n*',[case(_),phi([3,sg],K)],
        [phase(infl_inh,_),aB(mergeR(root,_))]) :- key(K).
lexicon('n*',[case(_),phi([3,pl],K)],
        [phase(infl_inh,_),aB(mergeR(root,_))]) :- key(K).
lexicon(c,[],[phase(infl_nom,_)]).
lexicon(tpast,[tns(past),infl(invis(_INFL))],[]).
lexicon(t,[tns(_),infl(invis(_INFL))],[]).
lexicon(d,[],[aB(mergeR(dRoot,_))]).
lexicon('\'s',[],[phase(infl_inh,_),aB(mergeR(n,_))]).
```

`phase(inf_acc,_)` is a function with bundle of features `infl_acc` that will be transmitted during ESM of the head

`aB(mergeR(root,_))` is an abbreviated (aB) function that applies `mergeR(root,_)` to merges with this head, recursively (until it is satisfied). `mergeR(root,_)`, *defined later*, means it must be merged with a root.

grammar.pl

Merge Abbreviations (aB)

A Computational Hack: abbreviations save memory

```
%%% Abbreviations

%% e.g. ipm *mergeR(int) SO: <{the,d},d>
mergeR(R,_) abbreviates function mergeR(sister1(X,selectType(X,R))).

mergeR(cat) abbreviates function
mergeR(upsister(X,isHead(X),categorizer(X))).

%% permits ipm SO: <{the,d},the>
mergeR(cat_d) abbreviates function
mergeR(upsister(X,isHead(X),categorizer_d(X))).
```

Example: n

grammar.pl

- looks up the type of X.
- succeeds if type of X = Type

```
%% called by mergeR(so(SO,selectType(SO,Type)))
selectType(X,Type) :-
        headOf(X,Hd), hdName(Hd,N),
        isType(N,Type),
        !.                              % green (headOf backtracking)
```

Example: n

grammar.pl

```
%% categorizers select for roots
categorizer(X) :-
        (X has_function aB(mergeR(root,_)) ->
         true
        ;
         X has_function mergeR(root,_)).

categorizer_d(X) :-
        (X has_function aB(mergeR(dRoot,_)) ->
         true
        ;
         X has_function mergeR(dRoot,_)).
```

Implementation of **mergeR**

```
%% External Set Merge (ESM)
%% applies mergeR Up to SO (or A or pass up)
%% extract mergeR from head A, apply to A (or SO or pass up)
%% extract mergeR from head SO, apply to SO (or A or pass up)
%% A a phase head, transmit INFL features
%% SO a phase, A a head, trigger transfer (after edge of SO formed)
%% (does A need be a head? ESM {XP,YP}.)
%% Special case (for now): label(_) attribute added when n* merged with {R,XP}
esm(SO,A,ESM,Result) :-
        ESM1 = {SOp,A}:esm,
        (active(merge_r) ->
         (applyUpMergeR(SO,A,Up1) ->
          append(Up1,Up2,Up3),
          (applyHdMergeR(A,SO,Up2) ->
           append(Up3,Up4,Up5),
           (applyHdMergeR(SO,A,Up4) ->
            append(Up5,Up6,Up7),
            (applyUpMergeR(A,SO,Up6) ->
             esm2(SO,A,SOp,Result),
             (Up7 == [] -> ESM = ESM1 ; ESM = ESM1:mR(Up7))
```

given SO and A, form {SO,A}

- applyUpMergeR: applies any surviving mergeR from SO (or A) to each other.
- applyHdMergeR: applies any head mergeR from SO (or A) to each other.

grammar.pl

documentation a bit out of data:
no upsister described here

```
%% merge X and Y
%% up(R) -> R on {X,Y}
%% up01(R) -> apply R now, if fail try up(R)
%% sister(SO,P) -> call(P) on Y, □ on {X,Y}
%% sister1(SO,P) like above except can only be applied once
%% so(SO,P) -> call(P) on X, □ on {X,Y}
%% so1(SO,P) like above except can only be applied once
%% so_s1(SO1,SO2,P) -> call(P) on X and Y, □ on {X,Y}
%% so_s1(SO1,SO2,P) like above except can only be applied once
%% abbreviations can be used: aB(Restriction)
%% Note: Flag comes from lexical entry: var or used
```

grammar.pl

Other kinds of abbreviations:
also used to save memory …

```
n3sg abbreviates lexicon (n$phi([3,sg],_)).
n3pl abbreviates lexicon (n$phi([3,pl],_)).
'n*3sg' abbreviates lexicon ('n*'$phi([3,sg],_)).
'n*3pl' abbreviates lexicon ('n*'$phi([3,pl],_)).

%% root(_) used if transmitting to a root
infl_nom abbreviates features [value(case(nom)),phi(_,_)].
infl_acc abbreviates features [value(case(acc)),phi(_,_),root(_)].
%% special case: no !phi for inherent Case
infl_inh abbreviates features [value(case(inh)),root(_)].
```

Abstract

A Relabeling-Analysis of English Possessives

Jason Ginsburg — Osaka Kyoiku University
Sandiway Fong — University of Arizona

English possessive DPs have been widely studied; e.g., see Bernstein and Tortora (B&R) (2005), Barker (1998), and references therein. Genitive possessive pronoun constructions in English are partially irregular, as exemplified in (1). Note that *of*-insertion is regular but there is variation in the deployment of the double genitive.

(1) a. my friend/the friend of mine/*the friend of mine's
b. your friend/*the friend of your/the friend of yours
c. his friend/the friend of his/*the friend of his'(s)
d. her friend/*the friend of her/the friend of hers
e. their friend/*the friend of their/the friend of theirs

We propose an account in the recent Minimalist framework of Chomsky (2013), extending Cechetto and Donati's (C&D) (2015) relabeling proposal for relative clauses.

In C&D, the term "relabeling" specifically refers to internal Merge of a noun to relabel a clause as a nominal, e.g. as in the free relative interpretation of "what you bought" in "I like what you bought", cf. "I wonder what you bought". (Chomsky's framework permits either α or β to contribute the label of {α,β} when two syntactic objects α and β Merge.) We propose that a PP is the target of relabeling in the relevant possessive pronoun examples in (1).

Our proposed structures are given in (2-6). We adopt a standard analysis of the DP in (2). In (2b), the determiner *the* bears unvalued Case (uCase) and labels the resulting phrase when Merged with the nominal *friend*. (We use DP, rather than D, for clarity of exposition, and assume Case is visible at the DP level.) We propose that (3a) receives the derivation shown in (3b-e). In (3b), following Chomsky (1986), we assume *'s* is a relational determiner that allows a DP to be Merged to its edge, cf. *John's friend*; also, *'s* values Case for the edge DP, with strikethrough marking valued Case. In (3c), the preposition *of* Merges and values Case for the complement DP, followed by internal Merge of *friend* – the relabeling step in (3d). We assume Merge is free; the impossibility of (4a) is predicted as the nominal *friend* in (4b) cannot value Case for DP *his friend*. In (3e), *the* is externally Merged, and we assume that an irregular spellout rule produces *his* from *he+'s*. In the regular case, e.g. *John's* in (5a) – assuming the derivation in (5b), the default rule for *'s* invokes no spellout change. However, as the data in (1) indicates, the presence of the pronominal double genitive cannot be predicted either syntactically or phonologically (see B&R). As pronouns are high-frequency words, we assume context-sensitive word-specific spellout rules override the generic rule to produce *hers* from *she+'s*, (also *yours* and *theirs*) but *mine* from *I+'s* (cf. *mine's). Spellout context-sensitivity is required to distinguish (6a) from (6b); i.e. the rule for pronoun+*'s* must take into account whether or not the complement of *'s* is a copy.

(2) a. the friend
b. [$_{DP}$ [$_{D}$ the] [$_{N}$ friend]]$_{uCase}$

(3) a. the friend of his
b. [$_{DP}$[$_{DP}$ he]~~$_{uCase}$~~[$_{D}$[$_{D}$'s][$_{N}$ friend]]]$_{uCase}$
c. [$_{PP}$[$_{P}$ of][$_{DP}$[$_{DP}$ he][$_{D}$[$_{D}$'s][$_{N}$ friend]]]~~$_{uCase}$~~] (Merge head *of*)
d. [$_{N}$[$_{N}$ friend][$_{PP}$[$_{P}$ of][$_{DP}$[$_{DP}$ he][$_{D}$[$_{D}$'s][$_{N}$ ~~friend~~]]]] (relabel)

Selected conference slides… ☞

ELSJ Spring Forum 2017
April 24, 2017
Meiji Gakuin University

A Relabeling Analysis of English Possessives*

Jason Ginsburg — Osaka Kyoiku University — jginsbur@cc.osaka-kyoiku.ac.jp
Sandiway Fong** — University of Arizona — sandiway@email.arizona.edu

*Supported by Japan Society for the Promotion of Science Grant-in-Aid for Scientific Research #16K02769

**Currently on leave at Kyoto University

Outline

- Data
- Core Assumptions
- Combinatorics (the model)
- Examples
- Conclusion

Data

English possessive DPs

- In English, *of*-insertion is regular but there is variation in the deployment of the double genitive.

(1)

a. my friend/the friend of mine/*the friend of mine's

b. your friend/*the friend of your/the friend of yours

c. his friend/the friend of his/*the friend of his'(s)

d. her friend/*the friend of her/the friend of hers

e. their friend/*the friend of their/the friend of theirs

English possessive DPs

How do possessives work?

(2) my friend

- Assume that this is a DP with head *'s*. D needs Case (has uCase), but D also checks uCase on *friend*

(3) the friend of mine

- *friend* is the underlying object of *my*
- Why isn't *friend* pronounced in object position?
- There is a possession-type relation between *my* and *friend*.
- Compare (4a-b) (cf. Barker 1998). In 4a, John owns the picture. In (4b), the picture is of John.

(4) a. a picture of John's hangs in the gallery

b. a picture of John hangs in the gallery

- Why don't you say any of these:

(5) a. *the friend of my's

b. *the friend of mine's

English possessive DPs

- There is variation in the deployment of double genitives

(6) *the friend of mine* vs. **the friend of mine's*

(7) **the friend of your* vs. *the friend of yours*

(8) *the friend of his* vs. **the friend of his's*

(9) **the friend of her* vs. *the friend of hers*

(10) **the friend of their* vs. *the friend of theirs*

- *Assume that PF rules are at work.*
  - For example, in (6) *mine* blocks *mine's, in (7) yours* blocks *your*
  - *my + 's + ~~friend~~ = mine* (*my + 's + ~~friend~~ ≠ mine's, my's*)

Core Assumptions

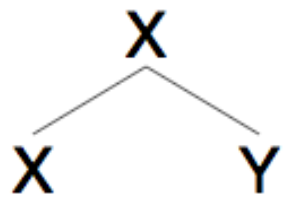
Proposal

- Theoretical underpinnings:
  - Minimalist framework of Chomsky (2013) – free Merge, labeling
  - Cecchetto and Donati's (2015) relabeling proposal (for relative clauses)
- Result:
  - We show how target examples (English possessives) can be **computed**

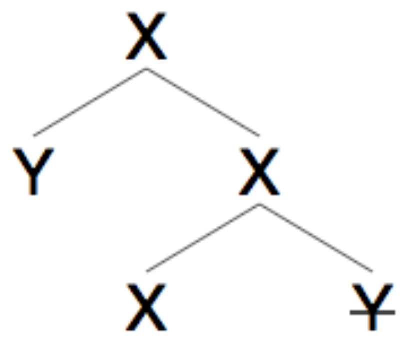
Merge

- Merge is free (Chomsky 2004, 2005, 2013, 2015)
  - no feature-driven movement
- Internal Merge (IM) and External Merge (EM) are free
  - IM and EM are both freely available*

External Merge of X with Y



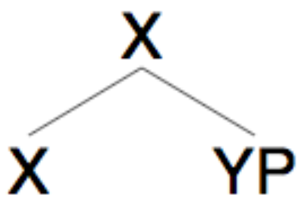
Internal Merge of Y with X



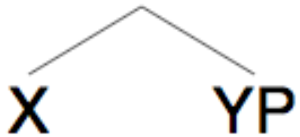
*A Chomsky 2017 lecture (University of Arizona) suggests IM is preferred over EM for minimal search reasons. Also see Shima (2000).

Set Merge: Labeling

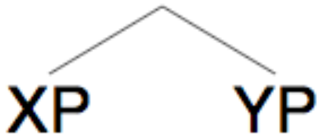
(a) Head X labels



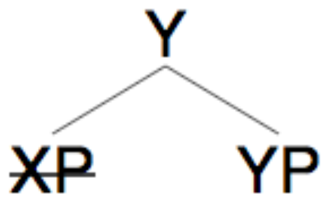
(b) Head X is too weak to label unless strengthened



(c) No label



(d) Y labels if XP moves out
- Not all copies of XP are within this Syntactic Object (SO)
- Y is not weak

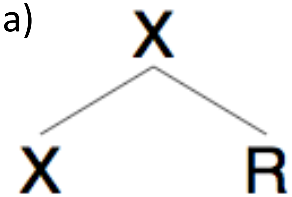


- External Set Merge is free
- Internal Set Merge is free

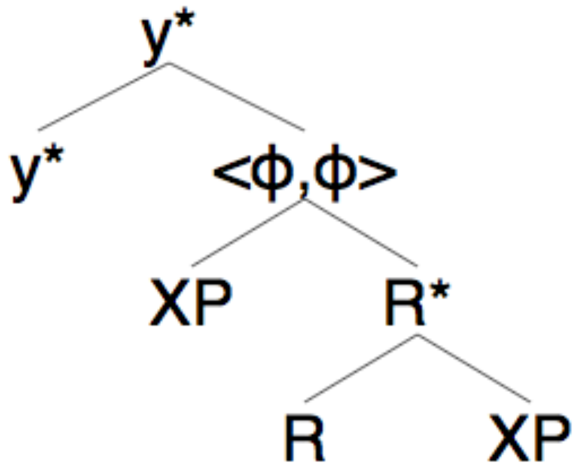
Strengthening

- R is weak
- In (a), categorizer X labels
- In (b), phase head y* transmits uPhi (and Case valuing) to R.
  - Agree(R,XP) checks uPhi on R
  - <ϕ,ϕ> labels, as R and XP have identical ϕ-features
  - strengthened R may label {R,XP} (* *represents strengthening*)
- In (c), n* strengthens R

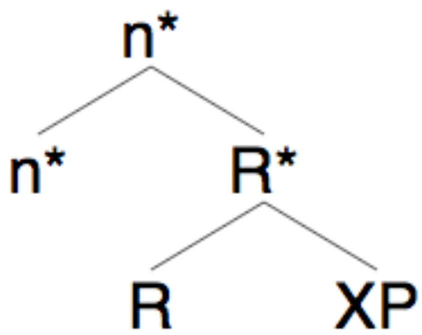
(a)



(b)



(c)



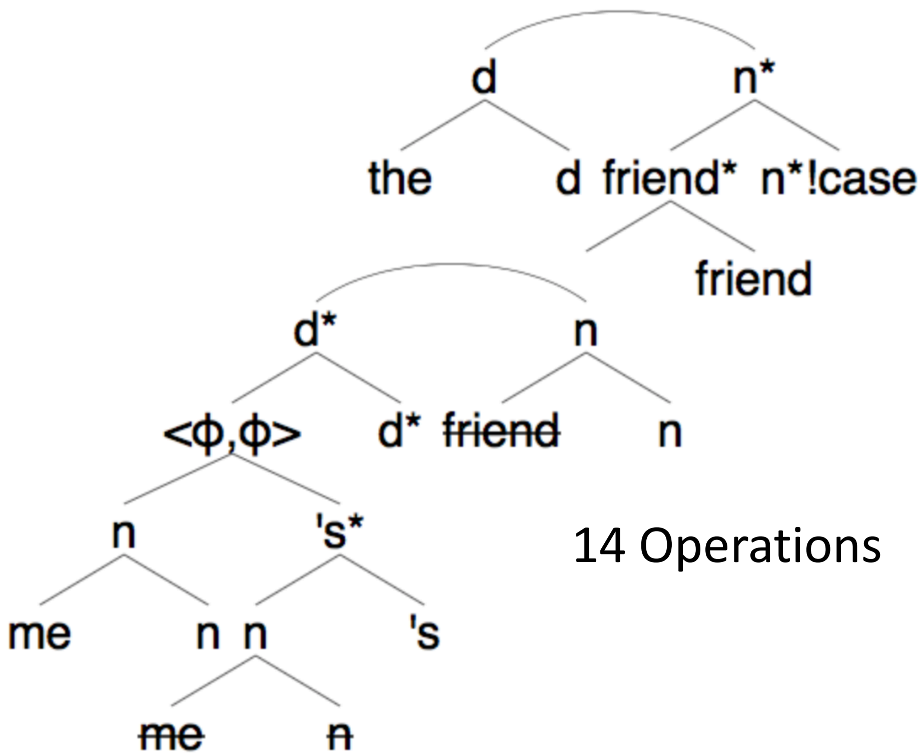
Combinatorics (the model)

Combinatorics: <{the,d},{book,n}>

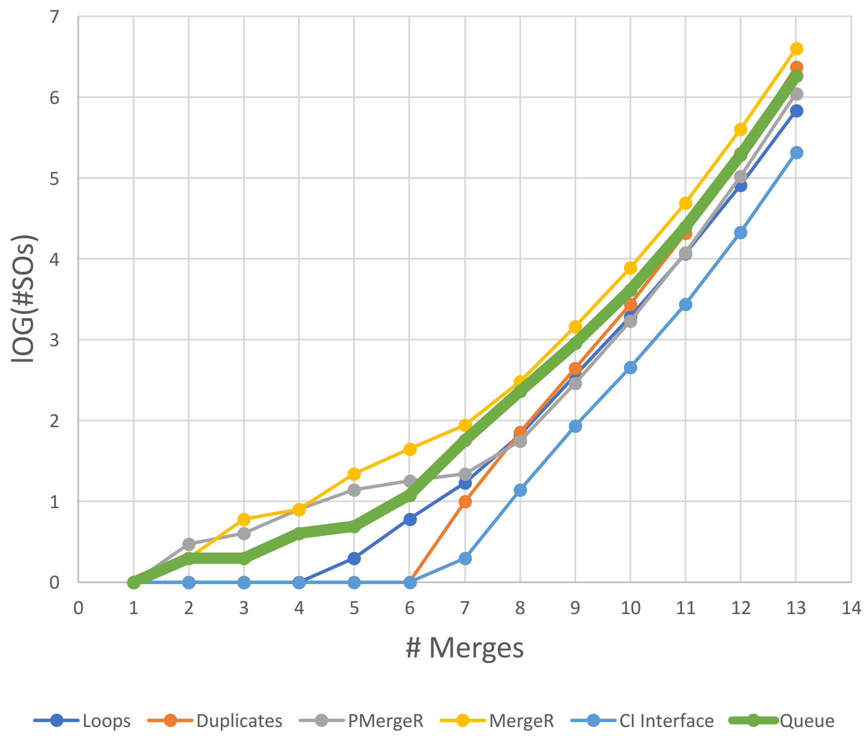
```
SO: book, Input: [n!case,[the,d]]
1 esm SO: {book,n!case}, Input: [[the,d]]
1 1 ism SO: {book,{book,n!case}}, Input: [[the,d]]
1 1 1 dws SO: the, Input: [d]
1 1 1 1 esm SO: {the,d}, Input: []
1 1 1 1 1 ipm SO: <{the,d},the>, Input: []
1 1 1 1 1 1 uws SO: {book,{book,n!case}}, Input: [<{the,d},the>]
1 1 1 1 1 1 1 epm *SO (unlabeled): <<{the,d},the>,{book,{book,n!case}}>, Input: []
1 1 1 1 1 1 2 ism SO: {{book,n!case},{book,{book,n!case}}}, Input: [<{the,d},the>]
1 1 1 1 1 1 2 1 epm *SO (unlabeled): <<{the,d},the>,{{book,n!case},{book,{book,n!case}}}>, Input: []
1 1 1 1 1 1 2 2 esm *SO (unlabeled): {{{book,n!case},{book,{book,n!case}}},<{the,d},the>}, Input: []
1 1 1 1 1 1 3 esm *SO (unlabeled): {{book,{book,n!case}},<{the,d},the>}, Input: []
1 1 1 1 2 ism SO: {the,{the,d}}, Input: []
1 1 1 1 2 1 uws SO: {book,{book,n!case}}, Input: [{the,{the,d}}]
1 1 1 1 2 1 1 epm *SO (unlabeled): <{the,{the,d}},{book,{book,n!case}}>, Input: []
1 1 1 1 2 1 2 ism SO: {{book,n!case},{book,{book,n!case}}}, Input: [{the,{the,d}}]
1 1 1 1 2 1 2 1 epm *SO (unlabeled): <{the,{the,d}},{{book,n!case},{book,{book,n!case}}}>, Input: []
1 1 1 1 2 1 2 2 esm *SO (unlabeled): {{{book,n!case},{book,{book,n!case}}},{the,{the,d}}}, Input: []
1 1 1 1 2 1 3 esm *SO (unlabeled): {{book,{book,n!case}},{the,{the,d}}}, Input: []
1 1 1 1 3 uws SO: {book,{book,n!case}}, Input: [{the,d}]
1 1 1 1 3 1 epm *SO (unlabeled): <{the,d},{book,{book,n!case}}>, Input: []
1 1 1 1 3 2 ism SO: {{book,n!case},{book,{book,n!case}}}, Input: [{the,d}]
1 1 1 1 3 2 1 epm *SO (unlabeled): <{the,d},{{book,n!case},{book,{book,n!case}}}>, Input: []
1 1 1 1 3 2 2 esm *SO (unlabeled): {{{book,n!case},{book,{book,n!case}}},{the,d}}, Input: []
1 1 1 1 3 3 esm *SO (unlabeled): {{book,{book,n!case}},{the,d}}, Input: []
1 2 dws SO: the, Input: [d]
1 2 1 esm SO: {the,d}, Input: []
1 2 1 1 ipm SO: <{the,d},the>, Input: []
1 2 1 1 1 uws SO: {book,n!case}, Input: [<{the,d},the>]
1 2 1 1 1 1 epm *SO (mergeR): <<{the,d},the>,{book,n!case}>, Input: []
1 2 1 1 1 2 esm *SO (unlabeled): {{book,n!case},<{the,d},the>}, Input: []
1 2 1 2 ism SO: {the,{the,d}}, Input: []
1 2 1 2 1 uws SO: {book,n!case}, Input: [{the,{the,d}}]
1 2 1 2 1 1 epm *SO (mergeR): <{the,{the,d}},{book,n!case}>, Input: []
1 2 1 2 1 2 esm *SO (unlabeled): {{book,n!case},{the,{the,d}}}, Input: []
1 2 1 3 uws SO: {book,n!case}, Input: [{the,d}]
1 2 1 3 1 epm *SO (end): <{the,d},{book,n!case}>, Input: []
1 2 1 3 2 ism SO: {book,{book,n!case}}, Input: [{the,d}]
1 2 1 3 2 1 epm *SO (unlabeled): <{the,d},{book,{book,n!case}}>, Input: []
1 2 1 3 2 2 esm *SO (unlabeled): {{book,{book,n!case}},{the,d}}, Input: []
1 2 1 3 3 esm *SO (unlabeled): {{book,n!case},{the,d}}, Input: []
```

- **Only convergent thread in the finite computation tree!**
- Statistics:
  - Nodes: 41
  - Loops detected: 13
  - Duplicates: 0
  - PMergeR: 15
  - MergeR: 53
  - Unlabeled: 17
  - Derivations: 1
  - Max # of merges: 8. Derivations completed.

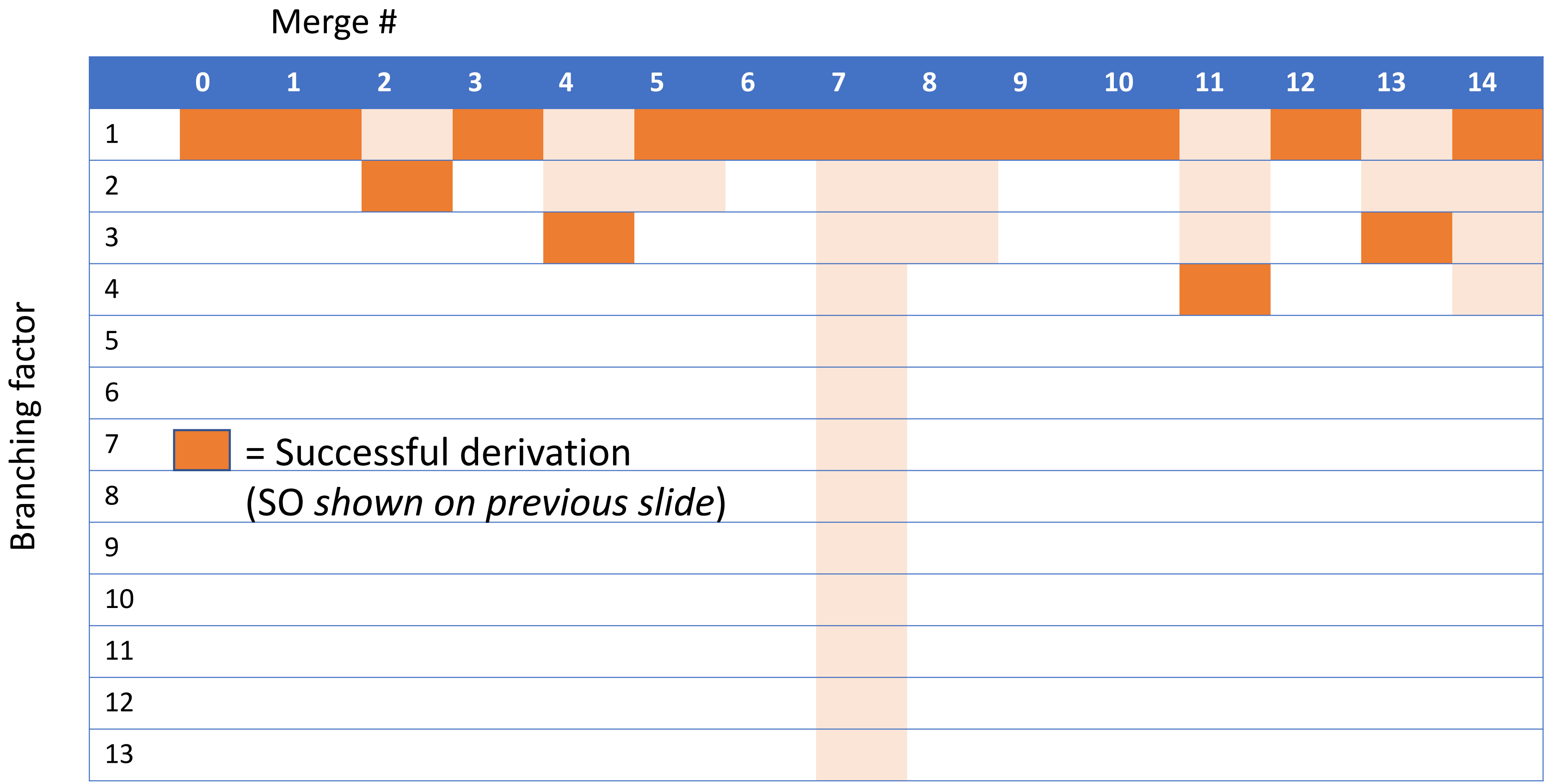
Combinatorics: *the friend of mine*



14 Operations



Combinatorics: *the friend of mine*



Heads: [friend,n!case,[me,n!case,'s,d*],n*!case,[the,d]]

Examples

Derivations

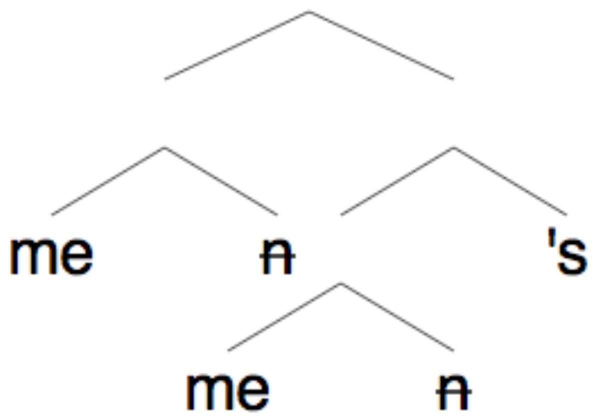
- SM and PM are free
- EM and IM are free
- Labeling occurs at the phase level
  - Complement of phase head is transferred
- Phase head transfers inflectional features to next lower head
- Rules of phonological form apply at transfer (after derivation is complete)
  - We assume no countercyclic operations (e.g. head-movement)
    - (some) head-movement phenomena can be relegated to Phonological Form (PF)

Derivations

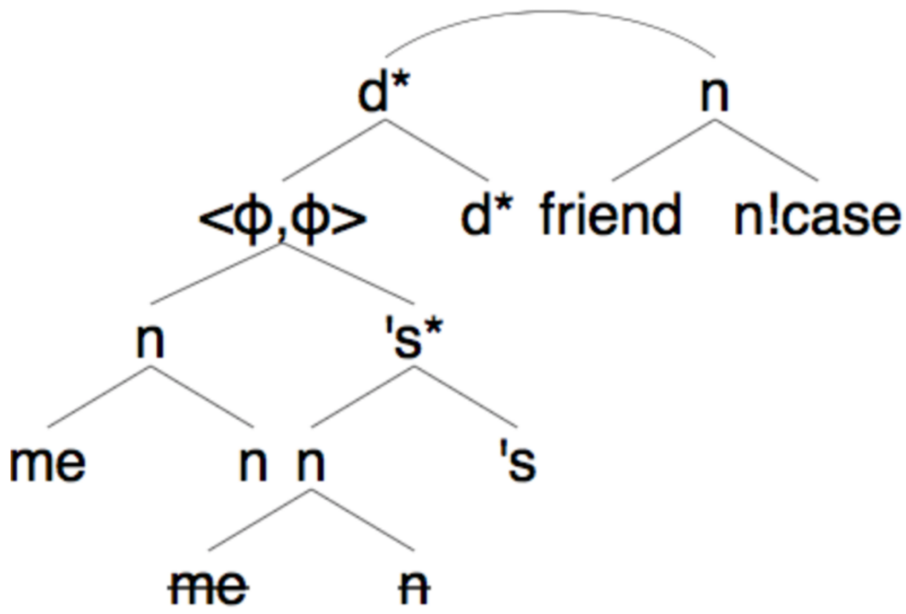
- Merge Restrictions:
  - (a) roots must be categorized (as soon as possible)
  - (b) each categorizer must find its root (with no intervening heads)
  - (c) categorizers can only categorize once
    - e.g. *{c,{R,{c,R}}} formed with only c and R (R=root, c=categorizer)
  - (d) can't PM β[uF] to α forming <β,α>, where β is an adjunct
    - since β is no longer accessible to operations, β[uF] can never get valued
- CI Interface Restriction:
  - (a) *<nP,nP>, *<dP,dP>
    - Interface expects <dP,nP>

my friend

- SM *me* (root) & *n*
- SM ‘s (root)
- SM (internal Set Merge) {me, n}

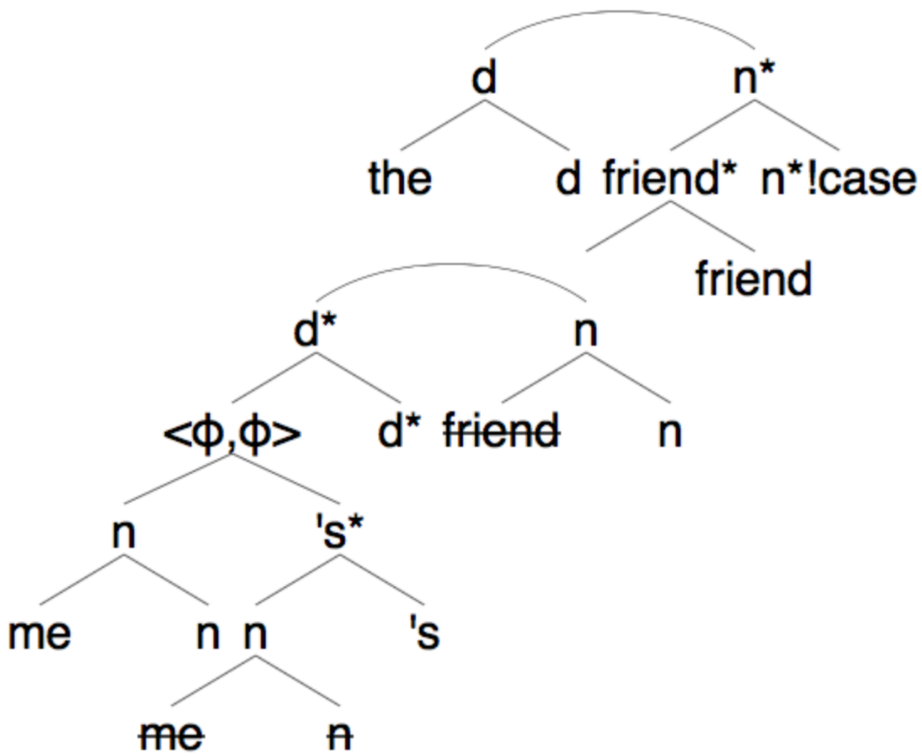


my friend



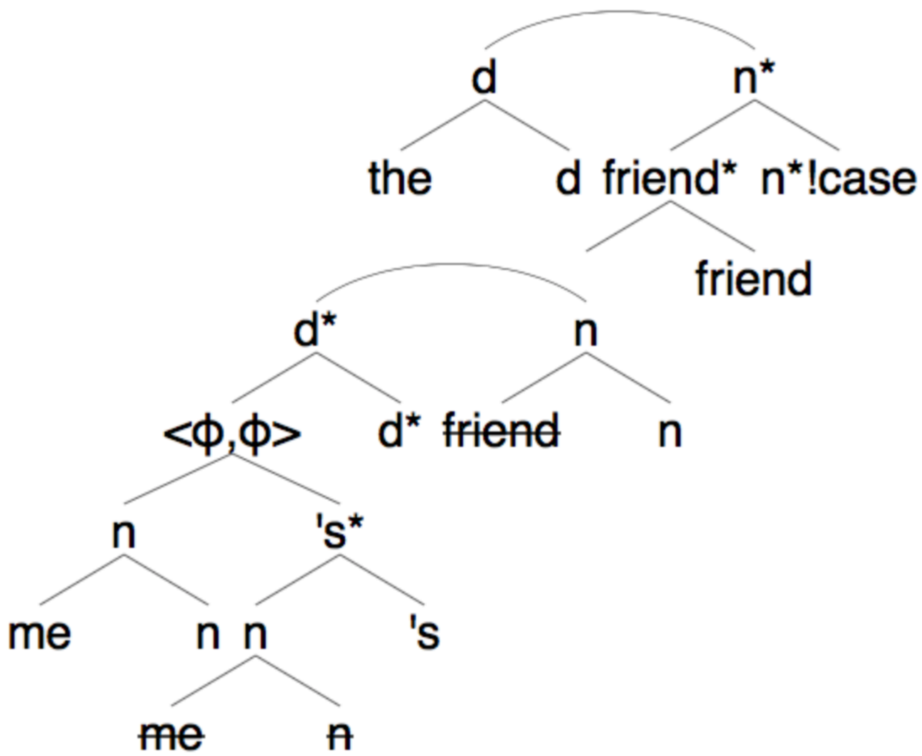
- SM *me* (root) & *n*
- SM ‘s (root)
- SM (internal Set Merge) {me, n}
- **SM d***
  - **uPhi and inherent Case are passed down to root ‘s**
- **Agree(‘s,n)**
  - **uPhi checked on ‘s**
  - **uCase checked on n**
- **d* (phase head) triggers transfer**
  - **Labeling occurs**
  - **Shared ϕ label**
    - **Shared ϕ strengthen ‘s → ‘s***
- **SM friend & n**
  - **!case = uninterpretable Case**
  - **n will label at transfer**
- **PM *my* & *friend***
- **Spell-Out**
  - **me ‘s friend → my friend**

the friend of mine



- form *my friend*
- PM *my friend* & *friend*
  - internal PM of *friend*
  - *my friend* = adjunct
- SM n*
- inherent Case is inherited by *friend*
- Agree(*friend,n*) → n gets inherent Case, pronounced as *of*
- SM d & *the*
- PM {*the,*d} & *friend of my*
  - {*the,*d} = adjunct

the friend of mine



- Spell-Out is tricky
  - Note that the elements of the tree aren't ordered yet
- {n, friend} has inherent case – spelled out as *of* preceding *my friend*
- *me* 's ~~friend n~~ = mine

the d friend InherentCase me 's ~~n friend~~ = the friend of mine

Conclusions

- An account of target possessive constructions in a featureless free-Merge system
- A computer model computes all possible structures (*perhaps a first*)
- However, overgeneration can cause issues:
- Examples:

(11) *the friend the

(12) *the friend of the

  - Perhaps assume that *the* (and *a*) can never be stranded in English

- Other puzzle:

(13) #Mary's friend of yours

  - on the intended reading: *Mary's friend* & *friend of yours*
  - a problem to be resolved past CI Interface's door?