Models of Adjunction in Minimalist Grammars

Thomas Graf
mail@thomasgraf.net
http://thomasgraf.net

Stony Brook University

FG 2014
August 17, 2014

The Theory-Neutral CliffsNotes

- Several properties set adjuncts apart from arguments.
- Which of these properties do recent proposals fail to capture, **and why**?
- Does linguistic adequacy increase formal complexity?

Insights

- **Empirical**
  Recursive adjunction poses biggest challenge
- **Formal**
  Optionality and iterability of adjuncts necessarily bring about a certain degree of complexity

Outline

1. Properties of Adjuncts

2. Three MG Models of Adjunction
   - A 1-Slide Intro to MGs
   - Category-Preserving Selection
   - Asymmetric Feature Checking
   - No Feature Checking

3. Formal Comparison

Properties of Adjuncts

Adjuncts are characterized by a variety of properties:

- optional
- iterable
- recursive adjunction
- ordering effects (only some adjuncts)
- no double adjunction
- adjuncts don’t project

Optionality

Grammaticality is preserved under removal of adjuncts.

(1) a. John **suddenly** abandoned his team.

b. John abandoned his team.

c. * John **suddenly** abandoned.

(2) a. John put the book **about Categorial Grammar** on the shelf.

b. John put the book on the shelf.

c. * John put the book **about Categorial Grammar**.

Iterability

The number of adjuncts per phrase is unbounded.

(3) a. the **terrible** destruction of the city

b. the **terrible unexpected** destruction of the city

c. * the **terrible** destruction of the city of the bridge

Recursivity

Adjuncts can be adjoined to.

(4) a. the **unexpected** destruction

b. the [**very unexpected**] destruction

c. the [**definitely** [**very unexpected**]] destruction

d. the [[**very definitely**] [**very unexpected**]] destruction

Ordering effects

Some adjuncts (in particular adjectives) exhibit a default word order. Deviating from this order often has semantic effects.

(5) a. the **big round** box

b. ? the **round big** box

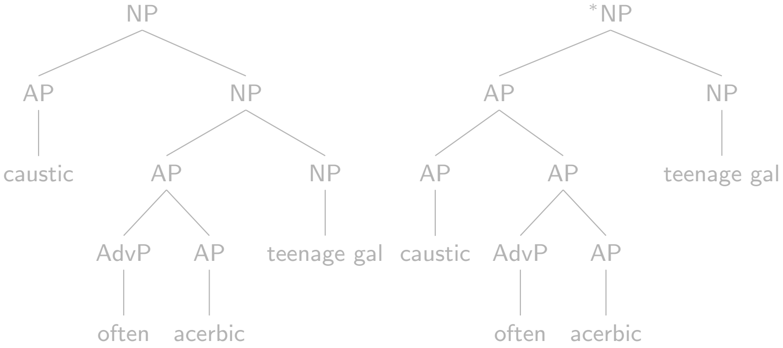
(6) a. a **beautiful old** clock

b. ? an **old beautiful** clock

No Double Adjunction

An adjunct adjoins to exactly one phrase.

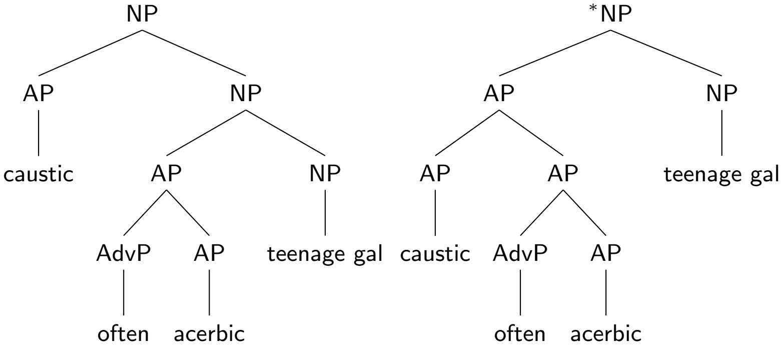
(7) the **caustic**, **often acerbic** teenage gal
$\neq$ the **often caustic**, **often acerbic** teenage gal



No Double Adjunction

An adjunct adjoins to exactly one phrase.

(7) the caustic, often acerbic teenage gal
≠ the often caustic, often acerbic teenage gal



Adjuncts Don't Project

Adjuncts are part of the phrase they adjoin to. At the same time, they occupy an "outer shell" compared to arguments.

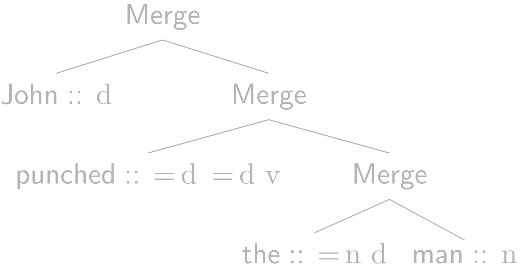
(8) a. John $[_{\text{VP}}$ $[_{\text{VP}}$ met Mary] **yesterday**], and Bill did ~~$[_{\text{VP}}$ $[_{\text{VP}}$ meet Mary] **yesterday**]~~, too.

b. John $[_{\text{VP}}$ $[_{\text{VP}}$ met Mary] **yesterday**], and Bill did $[_{\text{VP}}$ ~~$[_{\text{VP}}$ meet Mary]~~ **today**].

c. * John $[_{\text{VP}}$ met Mary], and Bill did $[_{\text{VP}}$ ~~meet~~ Sue].

Minimalist Grammars (Stabler 1997)

- Lexical items
  phonetic exponent :: *ordered list of features*
- Structure-building operations
  Merge: combine two trees in one
  Move: displace subtrees
- Operation must be triggered by features of opposite polarity



Merge

John :: d

Merge

punched :: =d =d v

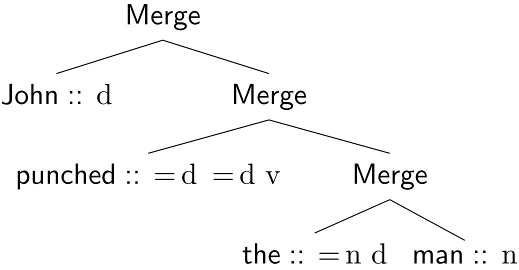
Merge

the :: =n d

man :: n

Minimalist Grammars (Stabler 1997)

- Lexical items
  phonetic exponent :: *ordered list of features*
- Structure-building operations
  Merge: combine two trees in one
  Move: displace subtrees
- Operation must be triggered by features of opposite polarity



Merge

John :: d

Merge

punched :: $=\mathrm{d}\ =\mathrm{d}\ \mathrm{v}$

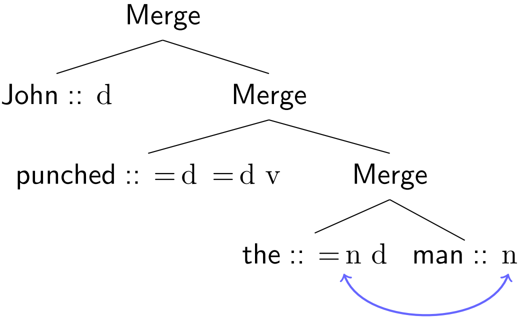
Merge

the :: $=\mathrm{n}\ \mathrm{d}$

man :: n

Minimalist Grammars (Stabler 1997)

- Lexical items
  phonetic exponent :: *ordered list of features*
- Structure-building operations
  Merge: combine two trees in one
  Move: displace subtrees
- Operation must be triggered by features of opposite polarity



Merge

John :: d

Merge

punched :: $=\mathrm{d}\ =\mathrm{d}\ \mathrm{v}$

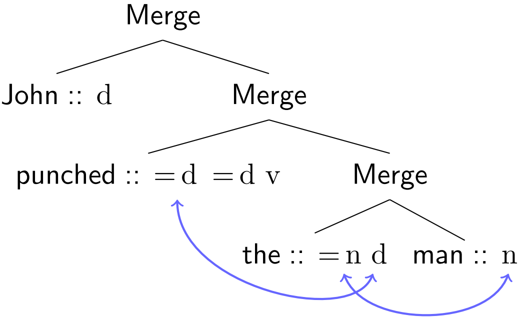
Merge

the :: $=\mathrm{n}\ \mathrm{d}$

man :: n

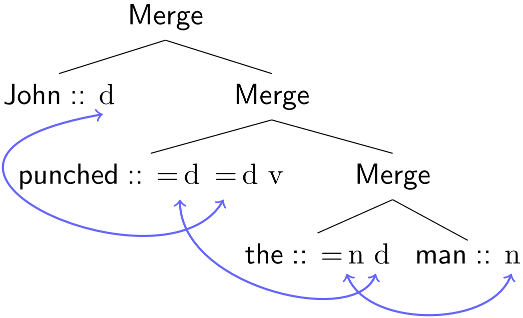
Minimalist Grammars (Stabler 1997)

- Lexical items
  phonetic exponent :: *ordered list of features*
- Structure-building operations
  Merge: combine two trees in one
  Move: displace subtrees
- Operation must be triggered by features of opposite polarity



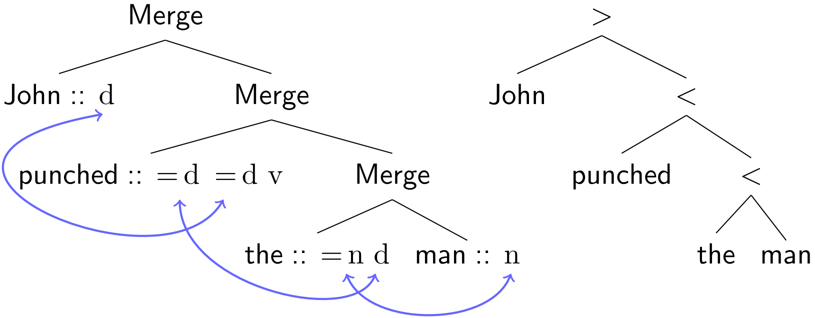
Minimalist Grammars (Stabler 1997)

- Lexical items
  phonetic exponent :: *ordered list of features*
- Structure-building operations
  Merge: combine two trees in one
  Move: displace subtrees
- Operation must be triggered by features of opposite polarity



Minimalist Grammars (Stabler 1997)

- Lexical items
  phonetic exponent :: *ordered list of features*
- Structure-building operations
  Merge: combine two trees in one
  Move: displace subtrees
- Operation must be triggered by features of opposite polarity


Merge
John :: d
Merge
punched :: $=\mathrm{d}\ =\mathrm{d}\ \mathrm{v}$
Merge
the :: $=\mathrm{n}\ \mathrm{d}$
man :: n
>
John
<
punched
<
the
man

Adjunction as Category-Preserving Selection (Folklore)

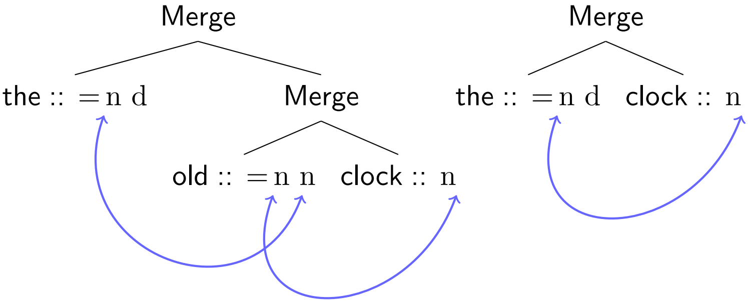
- Idea from CG: adjuncts have type τ/τ
- Adjuncts are just lexical items that happen to have category and selector features of the same name.

  $$\text{adjunct} :: \lambda \mathrm{x}[\ldots{=}\mathrm{x}\ldots\mathrm{x}\ldots](\mathrm{a})$$

- Advantage: no new machinery needed

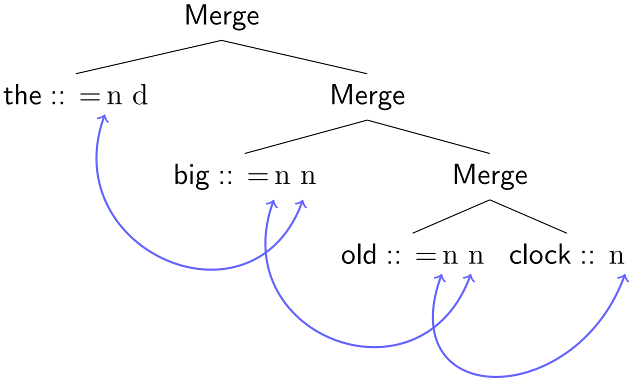
Success 1: Optionality

If *a* adjoins to *x*, *a* must have the same category feature as *x* ⇒ whatever selects *a* can also select *x* without *a*


Merge
the :: $=\mathrm{n}\ \mathrm{d}$
Merge
old :: $=\mathrm{n}\ \mathrm{n}$
clock :: n
Merge
the :: $=\mathrm{n}\ \mathrm{d}$
clock :: n

Success 2: Iterability

Since adjunction is category-preserving, whatever can adjoin to *x* can also adjoin to it after something else has already adjoined to *x*.

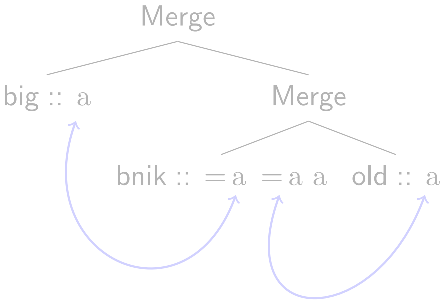

Merge
the :: $=\mathrm{n}\ \mathrm{d}$
Merge
big :: $=\mathrm{n}\ \mathrm{n}$
Merge
old :: $=\mathrm{n}\ \mathrm{n}$
clock :: n

Major Shortcomings

Treating adjunction as a special case of selection is **too restrictive and too permissive**.

Too Permissive: Double Adjunction

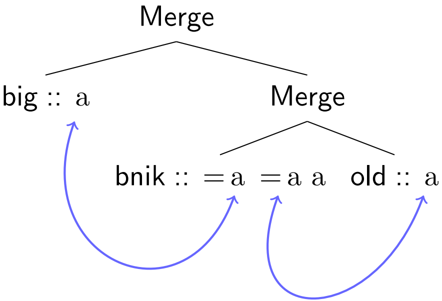
A lexical item like bnik :: $=\mathrm{a}\ =\mathrm{a}\ \mathrm{a}$ could be interpreted as an adjunct of two adjectives.



Major Shortcomings

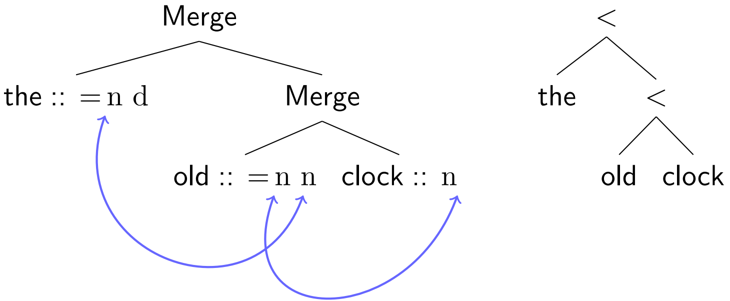
Treating adjunction as a special case of selection is **too restrictive and too permissive**.

Too Permissive: Double Adjunction
A lexical item like bnik :: $=\mathrm{a}\ \ =\mathrm{a}\ \mathrm{a}$ could be interpreted as an adjunct of two adjectives.



Too Restrictive: Incorrect Projection

Adjuncts **select** the phrase they adjoin to, hence adjuncts project.

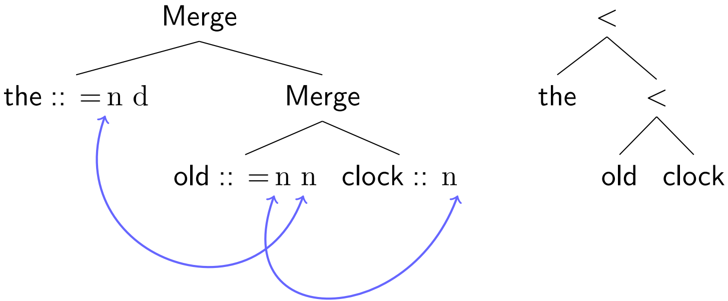


Why It Matters

Given how phrasal movement works in MG, this means that a moving XP leaves its adjuncts behind.

Too Restrictive: Incorrect Projection

Adjuncts **select** the phrase they adjoin to, hence adjuncts project.

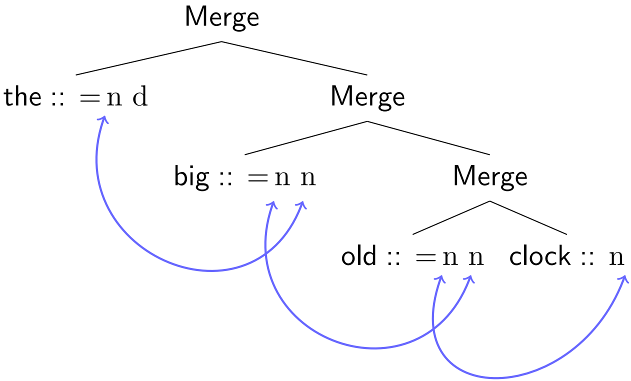


Why It Matters

Given how phrasal movement works in MG, this means that a moving XP leaves its adjuncts behind.

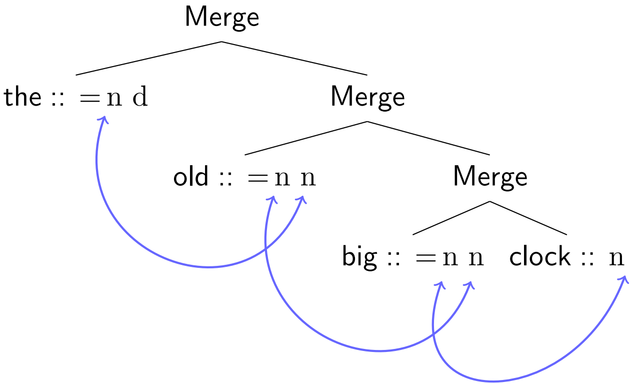
Too Restrictive: Ordering Effects

Ordering can easily be handled by standard selection,
but **not by category-preserving** selection = adjunction.



Too Restrictive: Ordering Effects

Ordering can easily be handled by standard selection,
but **not by category-preserving** selection = adjunction.

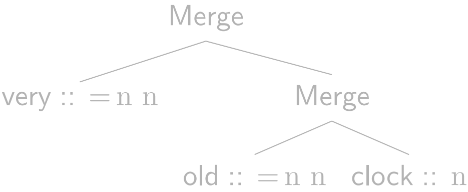

Merge
the :: $=\mathrm{n}\ \mathrm{d}$
Merge
old :: $=\mathrm{n}\ \mathrm{n}$
Merge
big :: $=\mathrm{n}\ \mathrm{n}$
clock :: n

Too Restrictive: Recursive Adjunction

Adjunction to an adjunct gives **wrong structure**.

- *clock* is a noun: clock :: n
- *old* modifies *clock*: old :: $=\mathrm{n}\ \mathrm{n}$
- *very* modifies *old*: very :: $=\mathrm{n}\ \mathrm{n}$

This produces a tree where *very* is an adjunct of *clock*, not *old*!



Merge

very :: $=\mathrm{n}\ \mathrm{n}$

Merge

old :: $=\mathrm{n}\ \mathrm{n}$

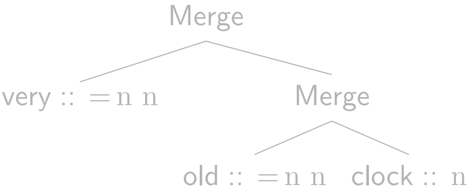
clock :: n

Too Restrictive: Recursive Adjunction

Adjunction to an adjunct gives **wrong structure**.

- *clock* is a noun: clock :: n
- *old* modifies *clock*: old :: $=\mathrm{n}\ \mathrm{n}$
- *very* modifies *old*: very :: $=\mathrm{n}\ \mathrm{n}$

This produces a tree where *very* is an adjunct of *clock*, not *old*!

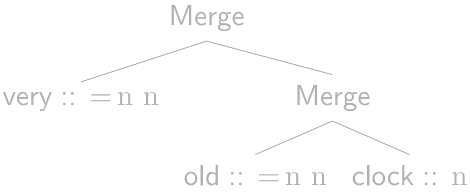

Merge
very :: $=\mathrm{n}\ \mathrm{n}$
Merge
old :: $=\mathrm{n}\ \mathrm{n}$
clock :: n

Too Restrictive: Recursive Adjunction

Adjunction to an adjunct gives **wrong structure**.

- *clock* is a noun: clock :: n
- *old* modifies *clock*: old :: $=\mathrm{n}\ \mathrm{n}$
- *very* modifies *old*: very :: $=\mathrm{n}\ \mathrm{n}$

This produces a tree where *very* is an adjunct of *clock*, not *old*!

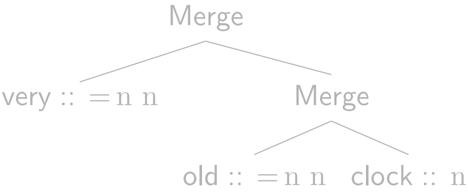


Too Restrictive: Recursive Adjunction

Adjunction to an adjunct gives **wrong structure**.

- *clock* is a noun: clock :: n
- *old* modifies *clock*: old :: $=\mathrm{n}\ \mathrm{n}$
- *very* modifies *old*: very :: $=\mathrm{n}\ \mathrm{n}$

This produces a tree where *very* is an adjunct of *clock*, not *old*!

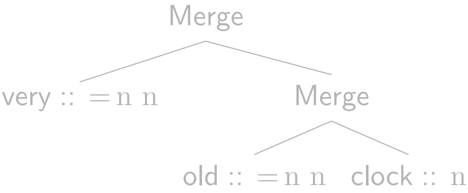

Merge
very :: $=\mathrm{n}\ \mathrm{n}$
Merge
old :: $=\mathrm{n}\ \mathrm{n}$
clock :: n

Too Restrictive: Recursive Adjunction

Adjunction to an adjunct gives **wrong structure**.

- *clock* is a noun: clock :: n
- *old* modifies *clock*: old :: $=\mathrm{n}\ \mathrm{n}$
- *very* modifies *old*: very :: $=\mathrm{n}\ \mathrm{n}$

This produces a tree where *very* is an adjunct of *clock*, not *old*!

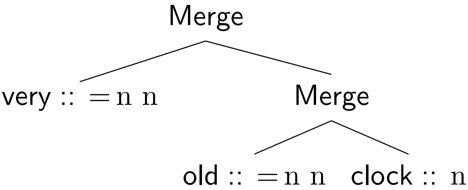

Merge
very :: $=\mathrm{n}\ \mathrm{n}$
Merge
old :: $=\mathrm{n}\ \mathrm{n}$
clock :: n

Too Restrictive: Recursive Adjunction

Adjunction to an adjunct gives **wrong structure**.

- *clock* is a noun: clock :: n
- *old* modifies *clock*: old :: $=\mathrm{n}\ \mathrm{n}$
- *very* modifies *old*: very :: $=\mathrm{n}\ \mathrm{n}$

This produces a tree where *very* is an adjunct of *clock*, not *old*!



Merge

very :: $=\mathrm{n}\ \mathrm{n}$

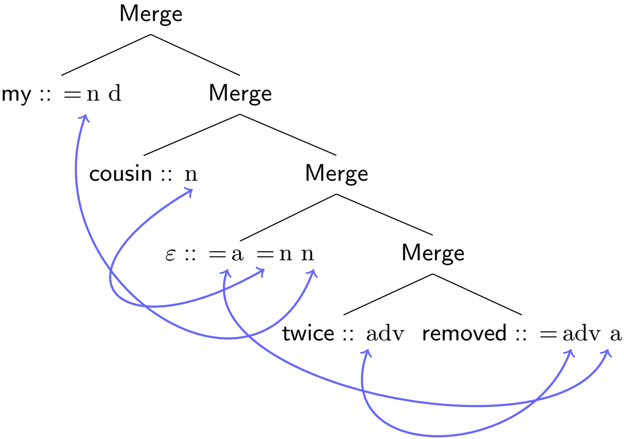
Merge

old :: $=\mathrm{n}\ \mathrm{n}$

clock :: n

The Real Problem: Adjuncts Need Special Status

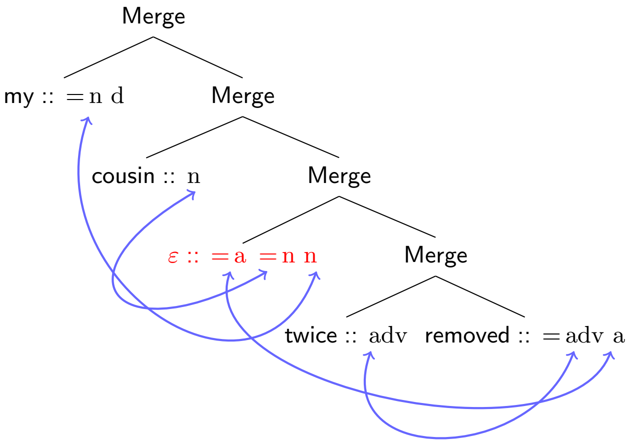
Fixing problems with coding tricks backfires:



The adjunct-marking empty head has an analogous feature type to possessive marker 's :: =n =d d , which is not an adjunct.

The Real Problem: Adjuncts Need Special Status

Fixing problems with coding tricks backfires:



The adjunct-marking empty head has an analogous feature type to possessive marker 's :: $=\mathrm{n}\ =\mathrm{d}\ \mathrm{d}$, which is not an adjunct.

Interim Summary

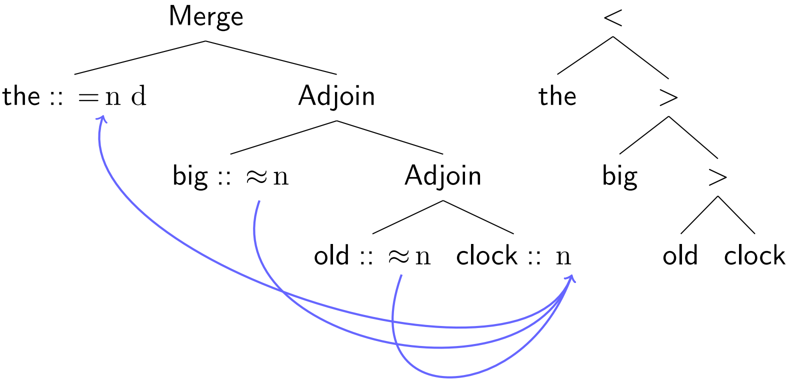
- Adjunction as category-preserving selection captures optionality and iterability.
- System must be relaxed to allow for ordering effects and recursive adjunction.
- A relaxed system can no longer distinguish adjuncts from arguments.

Conclusion

If we want to capture the properties of adjuncts, they need special status in the system.

Asymmetric Feature Checking (Frey and Gärtner 2002)

- Adjuncts have adjunction features instead of category features, e.g. old :: $\approx$n
- Adjunction features are checked by category feature of adjoined phrase, but not the other way round.
- By stipulation, adjuncts do not project.



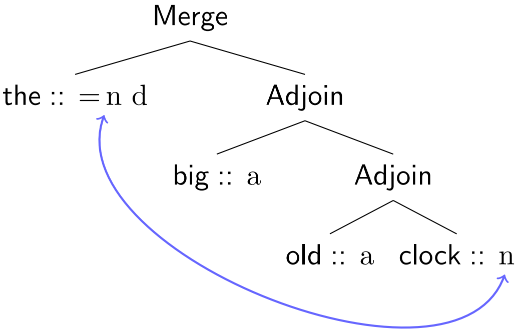
Evaluation

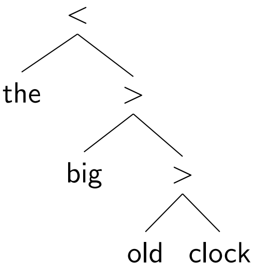
- **still captures**
  - optionality
  - iterability
- **also captures**
  - lack of projection (by stipulation)
  - lack of double adjunction
    adjunction feature must be checked exactly once
- **still fails**
  - ordering effects
  - recursive adjunction
    adjuncts have no category feature ⇒ cannot be adjoined to

No Feature Checking (Fowlie 2013)

- Adjuncts are freely inserted into derivation
- Relation $\mathcal{R}$ over category features determines whether adjunct may adjoin to phrase

$$\mathcal{R} : d > a > n$$

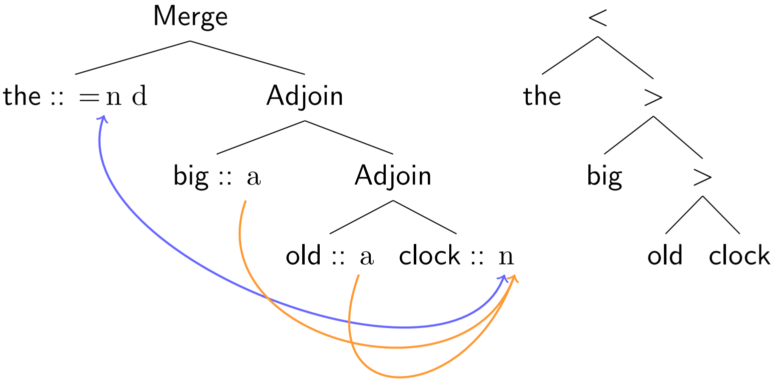




No Feature Checking (Fowlie 2013)

- Adjuncts are freely inserted into derivation
- Relation $\mathcal{R}$ over category features determines whether adjunct may adjoin to phrase

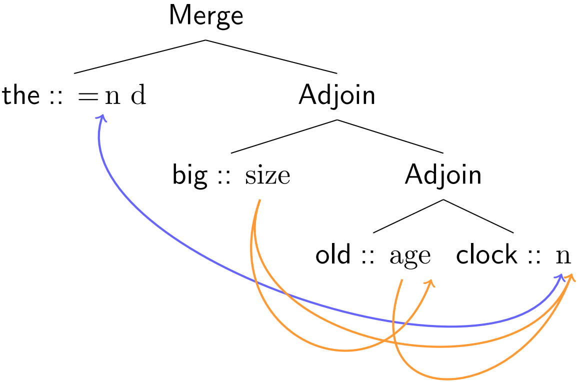
$\mathcal{R} : d > a > n$



Adding Order

All intervening adjuncts must also be lower on the hierarchy.

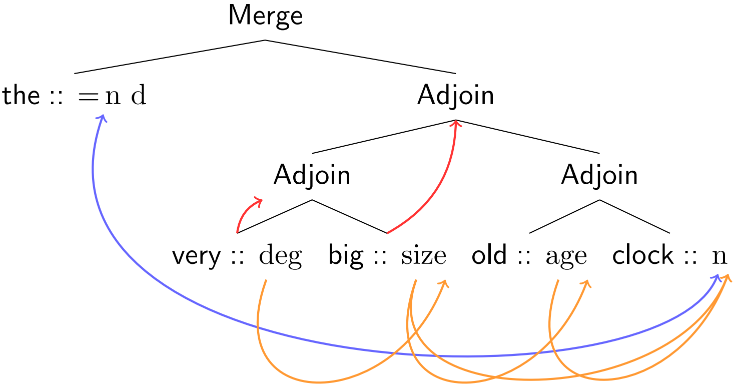
$\mathcal{R} : d > size > age > n$



Recursion

Adjunct need not be daughter of Adjoin node.

$$\mathcal{R} : d > size > age > n \cup deg > size > age$$



Summary of Linguistic Evaluation

	Cat. Preserv.	Asymm.	Free
optional	✓	✓	✓
iterable	✓	✓	✓
recursive	$\sim$	$\sim$	✓
no double adjunction		✓	✓
ordering effects	$\sim$	$\sim$	✓
correct projection		✓	✓

Overview of Formal Properties

- Formalisms are minor modifications of the model-theoretic definition of MGs as constraints over derivation trees (Graf 2012a,b, 2013)
- Complexity = complexity of derivation tree languages

	Merge	Cat.P	Asymm.	Free	Move
strictly local	✓				
vertical swap	✓				
homogeneous	✓				✓
FO[S]	✓				
FO[<]	✓				✓
reg ∩	✓				✓
gen. cap.	CFL				MCFL

Overview of Formal Properties

- Formalisms are minor modifications of the model-theoretic definition of MGs as constraints over derivation trees (Graf 2012a,b, 2013)
- Complexity = complexity of derivation tree languages

	Merge	Cat.P	Asymm.	Free	Move
strictly local	✓	✓	✓		
vertical swap	✓	✓	✓		
homogeneous	✓	✓	✓		✓
FO[S]	✓	✓	✓		
FO[<]	✓	✓	✓	✓	✓
reg ∩	✓				✓
gen. cap.	CFL	CFL	CFL	CFL	MCFL

No Closure Under Regular Intersection

- All implementations enforce the **optionality** of adjuncts.
- Let *L* be the regular language of trees *t* such that *t* contains at least one node labeled
  - very :: $=\mathrm{a}\ \mathrm{a}$, or
  - very :: $\approx\mathrm{a}$, or
  - very :: deg
- The intersection of *L* with MG *G*'s derivation tree language cannot be generated by any MG as every MG treats *very* as optional.

Moral O

Optionality of adjuncts is incompatible with closure under intersection with regular tree languages.

Non-Local Dependency

- Due to **iterability**, the distance between a head and the argument it selects is unbounded in the derivation tree.
- If the category of the argument can be inferred from the category of the adjuncts, it suffices to check the category or adjunction feature of the highest adjunct ⇒ local dependency
- But adjuncts are promiscuous (PP may adjoin to VP or NP)
  ⇒ must search for category of argument
  ⇒ long-distance dependency

Moral I

Iterability of adjuncts is incompatible with local selection unless the mapping from adjuncts to adjoinable categories is a function.

Summary

- Adjunction can be implemented in a variety of ways.
- Solution must be flexible to capture all properties of adjuncts, in particular
  - ordering
  - recursive adjunction
- Irrespective of the chosen implementation this entails:
  - no closure under regular intersection
  - selection is underlyingly a long-distance dependency

	Cat. Preserv.	Asymm.	Free
optional	✓	✓	✓
iterable	✓	✓	✓
recursive	∼	∼	✓
no double adjunction		✓	✓
ordering effects	∼	∼	✓
correct projection		✓	✓

References

Fowlie, Meaghan. 2013. Order and optionality: Minimalist grammars with adjunction. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, ed. András Kornai and Marco Kuhlmann, 12–20.

Frey, Werner, and Hans-Martin Gärtner. 2002. On the treatment of scrambling and adjunction in minimalist grammars. In *Proceedings of the Conference on Formal Grammar (FGTrento)*, 41–52. Trento.

Graf, Thomas. 2012a. Locality and the complexity of minimalist derivation tree languages. In *Formal Grammar 2010/2011*, ed. Philippe de Groot and Mark-Jan Nederhof, volume 7395 of *Lecture Notes in Computer Science*, 208–227. Heidelberg: Springer.

Graf, Thomas. 2012b. Movement-generalized minimalist grammars. In *LACL 2012*, ed. Denis Béchet and Alexander J. Dikovsky, volume 7351 of *Lecture Notes in Computer Science*, 58–73.

Graf, Thomas. 2013. *Local and transderivational constraints in syntax and semantics*. Doctoral Dissertation, UCLA.

Stabler, Edward P. 1997. Derivational minimalism. In *Logical aspects of computational linguistics*, ed. Christian Retoré, volume 1328 of *Lecture Notes in Computer Science*, 68–95. Berlin: Springer.