An Alternate View on Strong Lexicalization in TAG

Aniello De Santo, Alëna Aksënova and Thomas Graf

Stony Brook University
Department of Linguistics
aniello.desanto@stonybrook.edu

Düsseldorf, June 29 - July 1 2016
TAG+12

The Talk in a Nutshell

A well-known fact

Lexicalized grammars are *good* for parsing algorithms

Problem

TAGs not closed under strong lexicalization

Idea

- generalize TAGs to multi-dimensional TAGs;
- lexicalization via increase in dimensionality:
  ⇒ every d-TAG is strongly lexicalized by some $(d+1)$-TAG

The Talk in a Nutshell

A well-known fact

Lexicalized grammars are *good* for parsing algorithms

Problem

TAGs not closed under strong lexicalization

Idea

- generalize TAGs to multi-dimensional TAGs;
- lexicalization via increase in dimensionality:
  $\Rightarrow$ every d-TAG is strongly lexicalized by some $(d+1)$-TAG

The Talk in a Nutshell

A well-known fact

Lexicalized grammars are *good* for parsing algorithms

Problem

TAGs not closed under strong lexicalization

Idea

- generalize TAGs to multi-dimensional TAGs;
- lexicalization via increase in dimensionality:
  $\Rightarrow$ every d-TAG is strongly lexicalized by some (d+1)-TAG

Lexicalized Grammars

A grammar is lexicalized if the atoms from which compound structures are assembled each contain a pronounced lexical item.

Lexicalized grammars are *finitely ambiguous*:

- recognition is decidable;
- parsing is simplified [Schabes et al., 1988]

An Essential Distinction

weak lexicalization *vs* strong lexicalization

Lexicalized Grammars

A grammar is lexicalized if the atoms from which compound structures are assembled each contain a pronounced lexical item.

Lexicalized grammars are *finitely ambiguous*:

- recognition is decidable;
- parsing is simplified [Schabes et al., 1988]

An Essential Distinction

weak lexicalization *vs* strong lexicalization

TAGs and Lexicalization

Existing Results

- TAGs can be weakly lexicalized [Fujiyoshi, 2004]
- TAGs are not closed under strong lexicalization [Kuhlmann and Satta, 2012]
- TAGs are strongly lexicalized by context-free tree grammars of rank 2 [Maletti and Engelfriet, 2012]

Aim of this Paper

Derive lexicalization properties of TAGs by generalizing to multidimensional structures

- Every d-dimensional TAG is a $(d+1)$- dimensional TSG
- Every d-dimensional TSG is strongly lexicalized by some d-dimensional TAG
- $(d+1)$- TAGs strongly lexicalize d-TAGs

Existing Results

- TAGs can be weakly lexicalized [Fujiyoshi, 2004]
- TAGs are not closed under strong lexicalization [Kuhlmann and Satta, 2012]
- TAGs are strongly lexicalized by context-free tree grammars of rank 2 [Maletti and Engelfriet, 2012]

Aim of this Paper

Derive lexicalization properties of TAGs by generalizing to multidimensional structures

- Every d-dimensional TAG is a $(d+1)$- dimensional TSG
- Every d-dimensional TSG is strongly lexicalized by some d-dimensional TAG
- $(d+1)$- TAGs strongly lexicalize d-TAGs

TAGs and Lexicalization

Existing Results

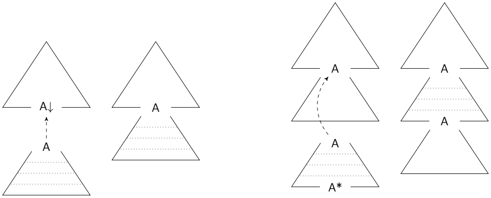
- TAGs can be weakly lexicalized [Fujiyoshi, 2004]
- TAGs are not closed under strong lexicalization [Kuhlmann and Satta, 2012]
- TAGs are strongly lexicalized by context-free tree grammars of rank 2 [Maletti and Engelfriet, 2012]

Aim of this Paper

Derive lexicalization properties of TAGs by generalizing to multidimensional structures

- Every d-dimensional TAG is a $(d+1)$- dimensional TSG
- Every d-dimensional TSG is strongly lexicalized by some d-dimensional TAG
- $(d+1)$- TAGs strongly lexicalize d-TAGs

Adjunction & Substitution

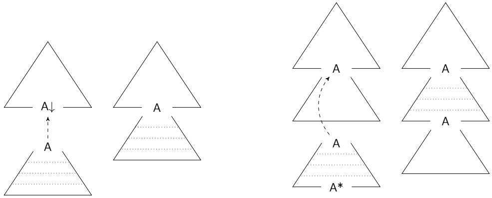


Substitution can be regarded as adjunction of a footless tree at a leaf node

Tree Substitution Grammar (TSG)

A restricted TAG where all licit instances of adjunction only rewrite leaf nodes

Adjunction & Substitution

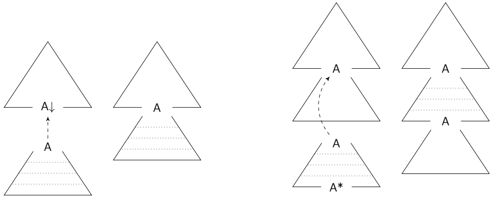


Substitution can be regarded as adjunction of a footless tree at a leaf node

Tree Substitution Grammar (TSG)

A restricted TAG where all licit instances of adjunction only rewrite leaf nodes

Adjunction & Substitution

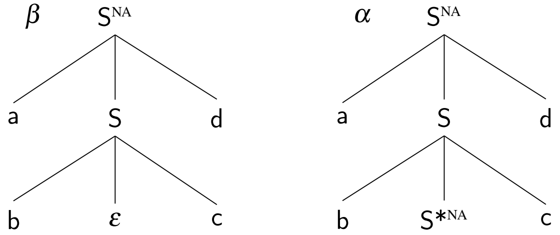


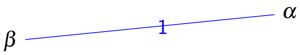
Substitution can be regarded as adjunction of a footless tree at a leaf node

Tree Substitution Grammar (TSG)

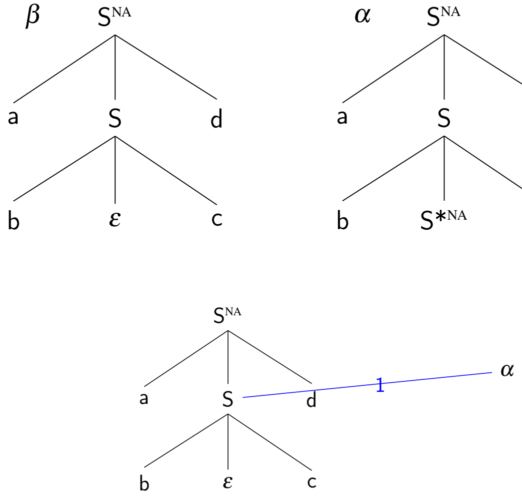
A restricted TAG where all licit instances of adjunction only rewrite leaf nodes

TAGs as 3-d trees [Rogers, 1998]

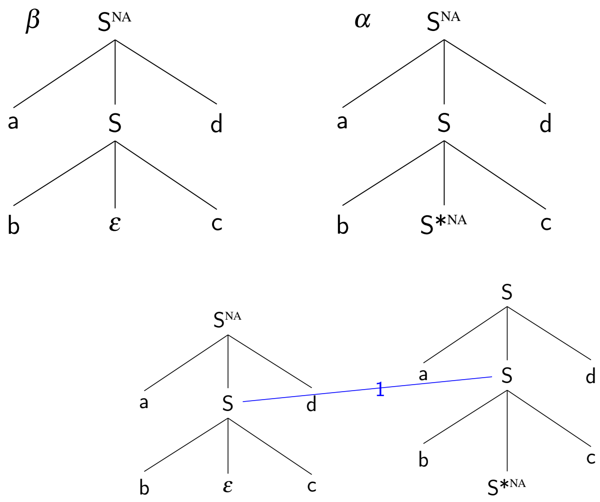




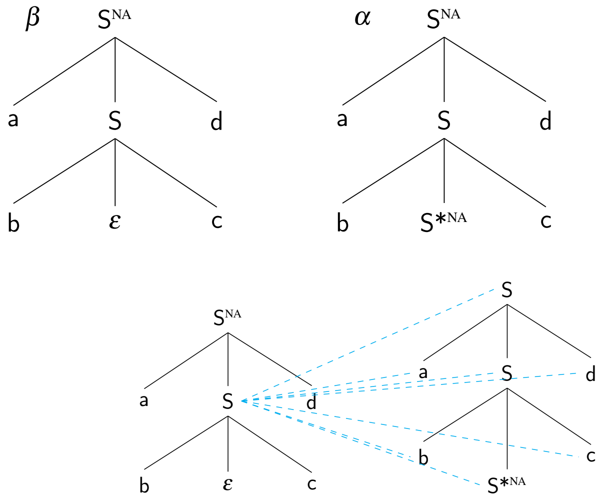
TAGs as 3-d trees [Rogers, 1998]



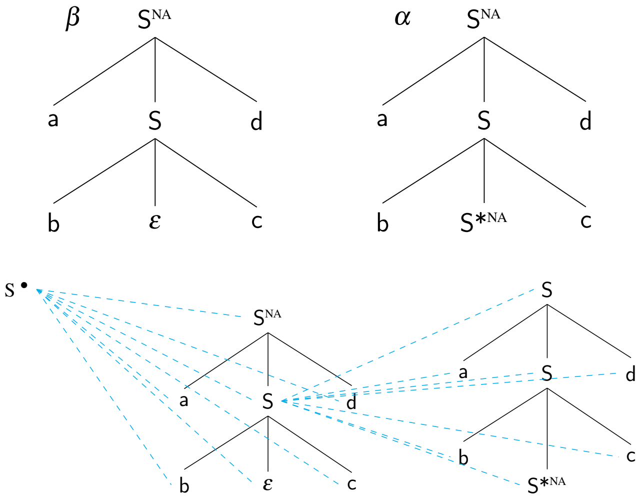
TAGs as 3-d trees [Rogers, 1998]



TAGs as 3-d trees [Rogers, 1998]

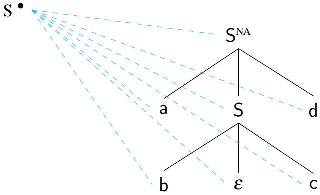


TAGs as 3-d trees [Rogers, 1998]



TAGs as multi-dimensional structures [Rogers, 2003]

We can increase the dimensionality of a grammar

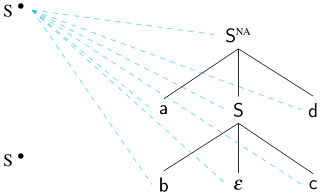


d-dimensional Local Structure

- d-dimensional mother
- yd^{d-1}: (d-1)-yield

TAGs as multi-dimensional structures [Rogers, 2003]

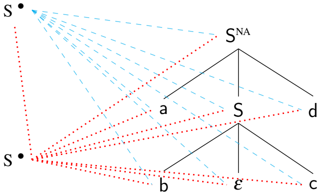
We can increase the dimensionality of a grammar



d-dimensional Local Structure

- d-dimensional mother
- yd^{d-1}: (d-1)-yield

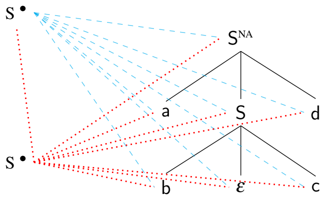
We can increase the dimensionality of a grammar



d-dimensional Local Structure

- d-dimensional mother
- yd^{d-1}: (d-1)-yield

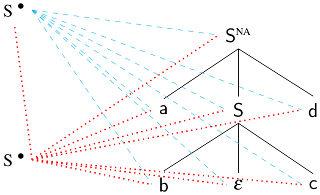
We can increase the dimensionality of a grammar



d-dimensional Local Structure

- d-dimensional mother
- yd^{d-1}: (d-1)-yield

We can increase the dimensionality of a grammar

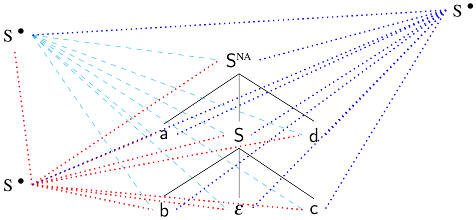


d-dimensional Local Structure

- d-dimensional mother
- yd^{d-1}: (d-1)-yield

TAGs as multi-dimensional structures [Rogers, 2003]

We can increase the dimensionality of a grammar



d-dimensional Local Structure

- d-dimensional mother
- yd^{d-1}: (d-1)-yield

A 4d Example

The 8-language

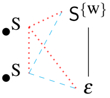
$$a^n b^n c^n d^n e^n f^n g^n h^n$$

(u) (w)

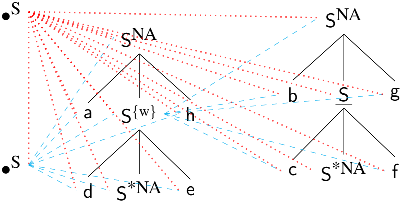
A 4d Example

The 8-language

$$a^n b^n c^n d^n e^n f^n g^n h^n$$



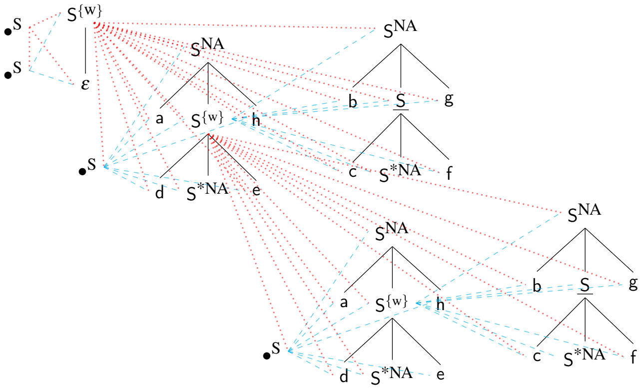
(u)



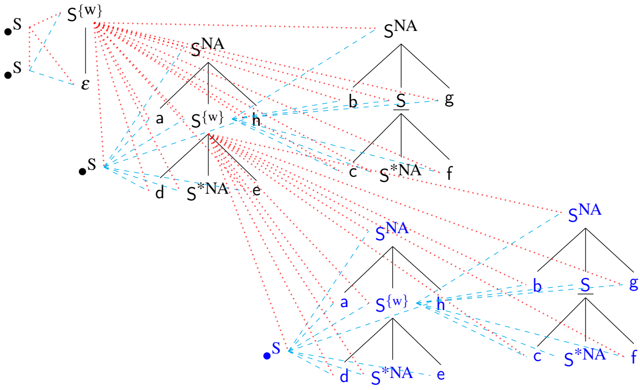
(w)

The 8-language: a derivation

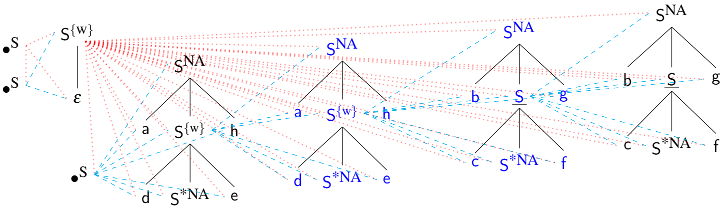
The 4-d structure



The 4-d structure

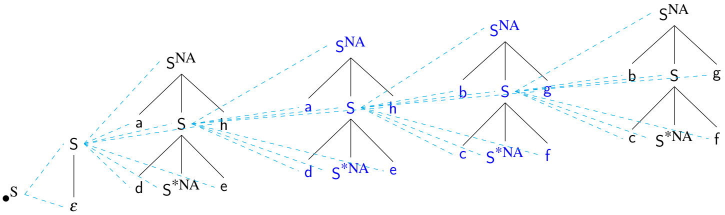


The 3-d yield ...

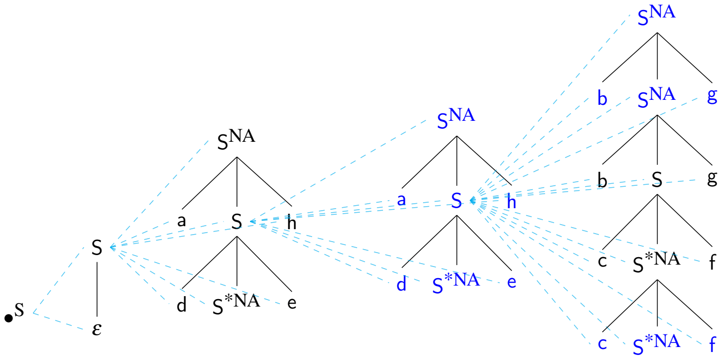

S
S
S{w}
ε
S
S^NA
a
S{w}
h
d
S*NA
e
S^NA
a
S{w}
h
d
S*NA
e
S^NA
b
S
g
c
S*NA
f
S^NA
b
S
g
c
S*NA
f

The 8-language: a derivation

The 3-d yield!

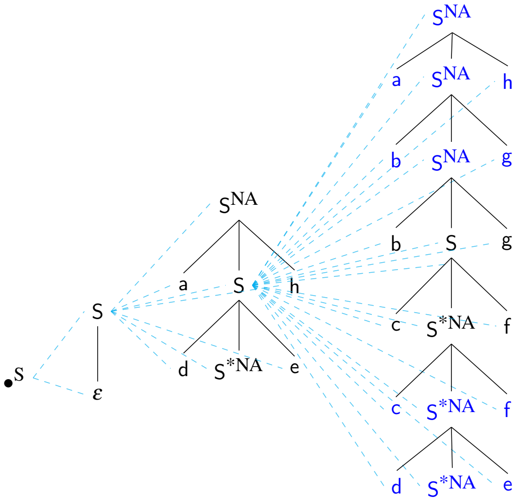


The 2-d yield ...



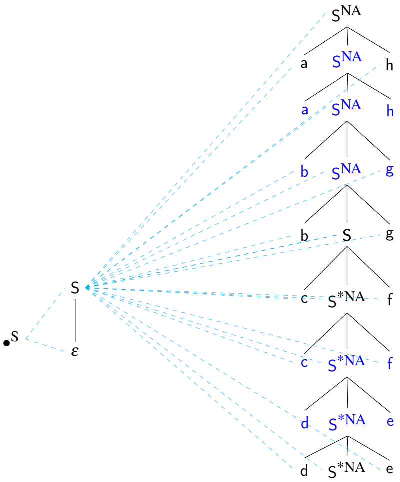
The 8-language: a derivation

The 2-d yield ...



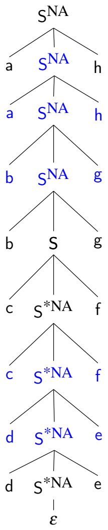
The 8-language: a derivation

The 2-d yield ...



The 8-language: a derivation

The 2-d yield!



Interim Summary

The Road so far

- Substitution as Adjunction
- TAGs as natural 3-d structures
- the generalization to higher dimensions is easy

Next Steps

We can generalize existing proofs to multidimensional structures:

- d-TAGs are $(d+1)$-TSGs
- d-TAGs strongly lexicalize d-TSGs
- $(d+1)$-TAGs strongly lexicalize d-TAGs

Interim Summary

The Road so far

- Substitution as Adjunction
- TAGs as natural 3-d structures
- the generalization to higher dimensions is easy

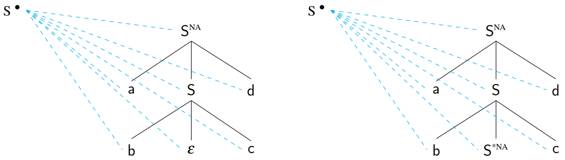
Next Steps

We can generalize existing proofs to multidimensional structures:

- d-TAGs are $(d+1)$-TSGs
- d-TAGs strongly lexicalize d-TSGs
- $(d+1)$-TAGs strongly lexicalize d-TAGs

d-TAGs are (d+1)-TSGs

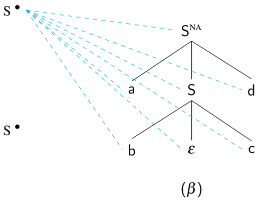
We can easily convert a 3d grammar into a 4d one



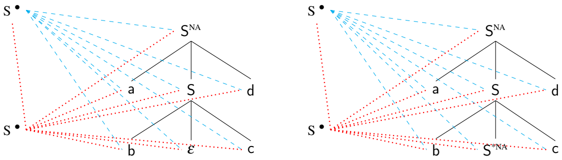
(β) (α)

d-TAGs are (d+1)-TSGs

We can easily convert a 3d grammar into a 4d one



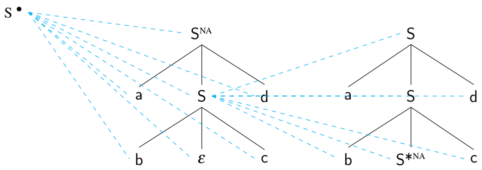
We can easily convert a 3d grammar into a 4d one



(β) (α)

d-TAGs are $(d+1)$-TSGs

We can show that adjunction in d is substitution in $(d+1)$

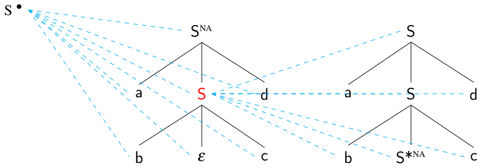


Properties of S

2-dimensional mother ⇒ 3d adjunction

d-TAGs are $(d+1)$-TSGs

We can show that adjunction in d is substitution in $(d+1)$

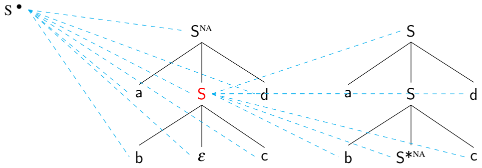


Properties of S

2-dimensional mother ⇒ 3d adjunction

d-TAGs are (d+1)-TSGs

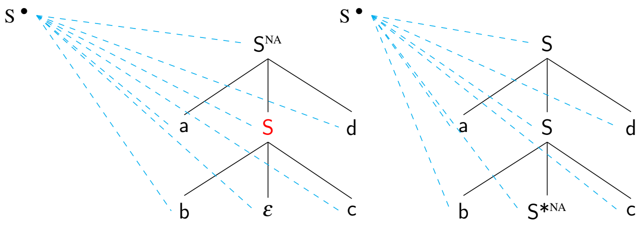
We can show that adjunction in d is substitution in (d+1)



Properties of S

2-dimensional mother ⇒ 3d adjunction

d-TAGs are (d+1)-TSGs



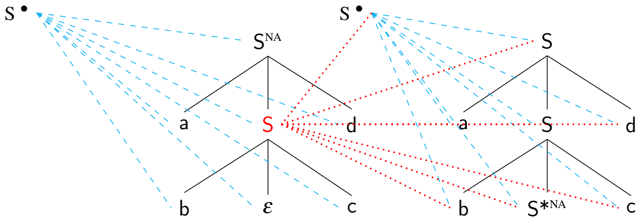
Properties of S

3-dimensional leaf ⇒ 4d substitution

Reminder: Tree Substitution Grammar (TSG)

A restricted TAG where all licit instances of adjunction only rewrite nodes that are not mothers in the (d-1)-dimension

d-TAGs are $(d+1)$-TSGs



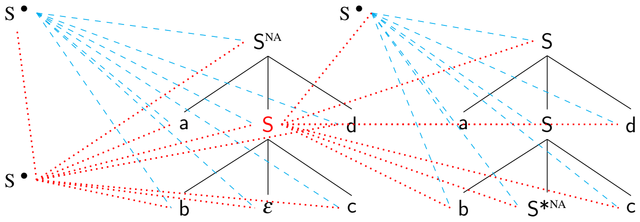
Properties of S

3-dimensional leaf ⇒ 4d substitution

Reminder: Tree Substitution Grammar (TSG)

A restricted TAG where all licit instances of adjunction only rewrite nodes that are not mothers in the (d-1)-dimension

d-TAGs are $(d+1)$-TSGs



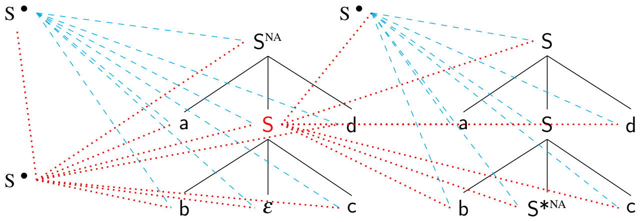
Properties of S

3-dimensional leaf ⇒ 4d substitution

Reminder: Tree Substitution Grammar (TSG)

A restricted TAG where all licit instances of adjunction only rewrite nodes that are not mothers in the (d-1)-dimension

d-TAGs are (d+1)-TSGs



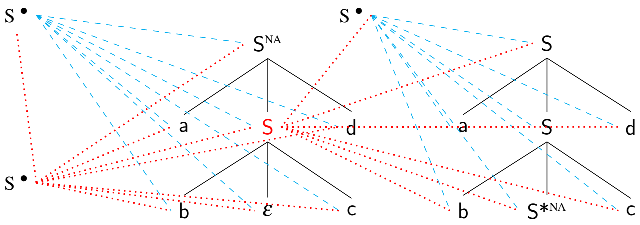
Properties of S

3-dimensional leaf ⇒ 4d substitution

Reminder: Tree Substitution Grammar (TSG)

A restricted TAG where all licit instances of adjunction only rewrite nodes that are not mothers in the (d-1)-dimension

d-TAGs are (d+1)-TSGs



Properties of S

3-dimensional leaf ⇒ 4d substitution

Reminder: Tree Substitution Grammar (TSG)

A restricted TAG where all licit instances of adjunction only rewrite nodes that are not mothers in the (d-1)-dimension

[Schabes, 1990]

TAGs strongly lexicalize TSGs.

A Lexicalization Procedure

Consider a TSG G:

1. Divide G in recursive and non-recursive;
2. Construct the set I_{lex} of initial trees;
3. Construct the set A of auxiliary trees.

We can extend the procedure to d-dimensional grammars

d-TAGs strongly lexicalize d-TSGs

[Schabes, 1990]

TAGs strongly lexicalize TSGs.

A Lexicalization Procedure

Consider a TSG G:

1. Divide G in recursive and non-recursive;
2. Construct the set I_{lex} of initial trees;
3. Construct the set A of auxiliary trees.

We can extend the procedure to d-dimensional grammars

d-TAGs strongly lexicalize d-TSGs

[Schabes, 1990]

TAGs strongly lexicalize TSGs.

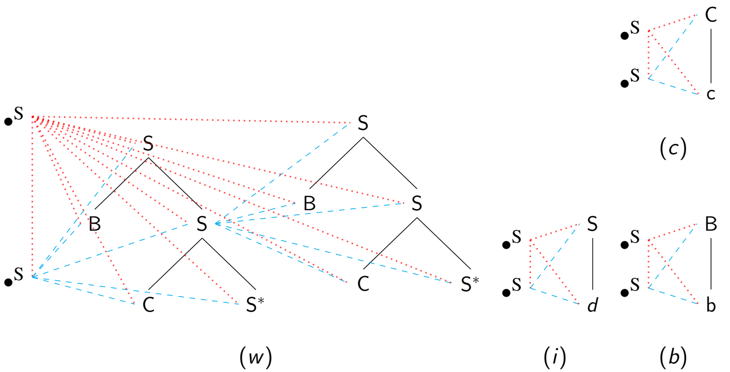
A Lexicalization Procedure

Consider a TSG G:

1. Divide G in recursive and non-recursive;
2. Construct the set I_{lex} of initial trees;
3. Construct the set A of auxiliary trees.

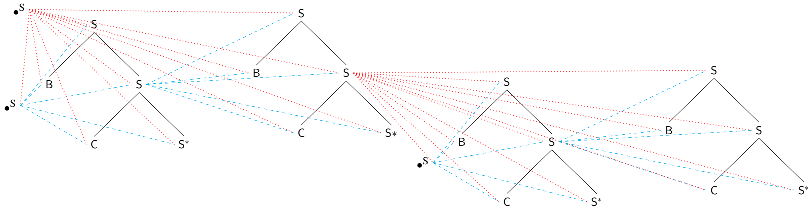
We can extend the procedure to d-dimensional grammars

A non lexicalized 4d-TSG

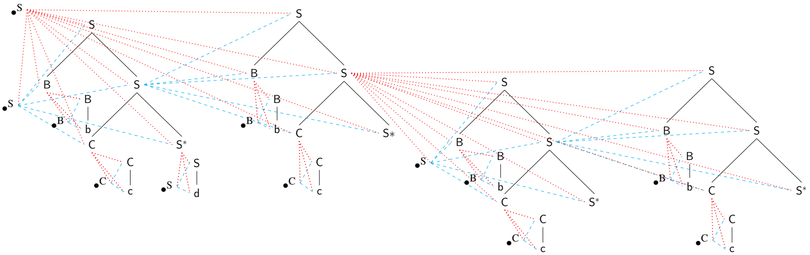


A non lexicalized 4d-TSG: a derivation

Substitution in 4d

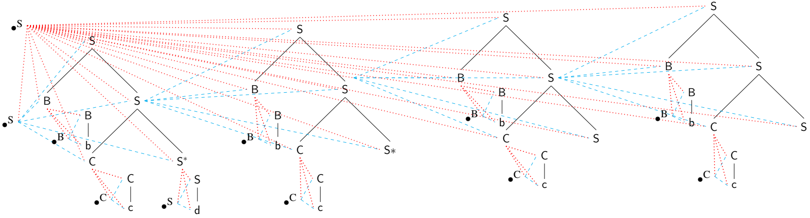


Substitution in 4d



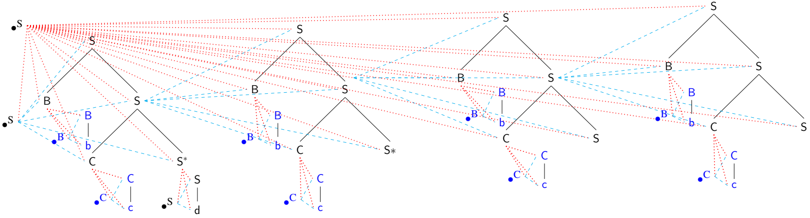
A non lexicalized 4d-TSG: a derivation

The 3d yield...



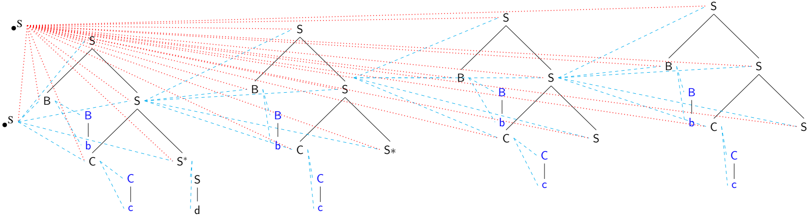
A non lexicalized 4d-TSG: a derivation

The 3d yield...



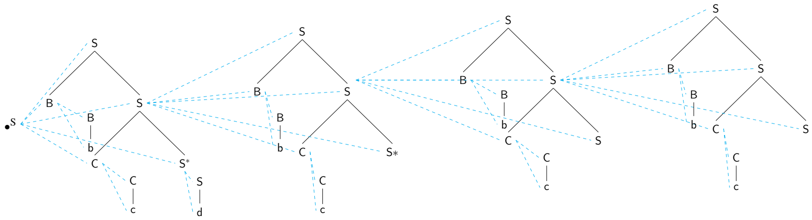
A non lexicalized 4d-TSG: a derivation

The 3d yield...



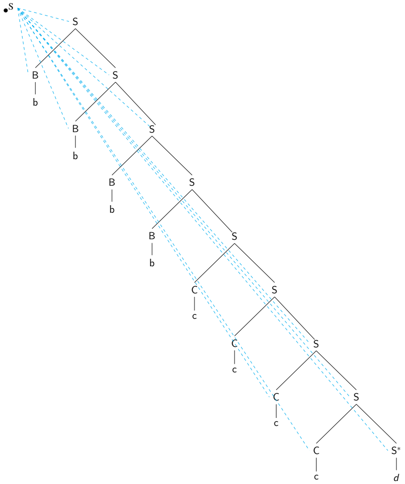
A non lexicalized 4d-TSG: a derivation

The 3d yield!



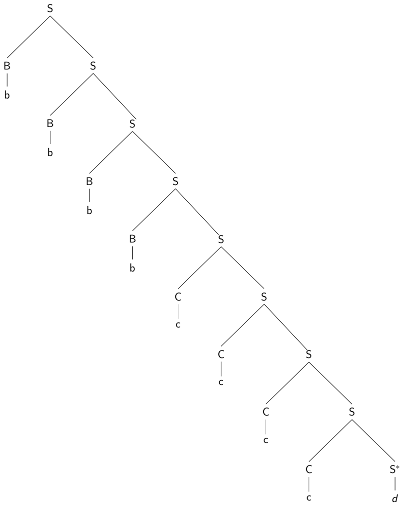
A non lexicalized 4d-TSG: a derivation

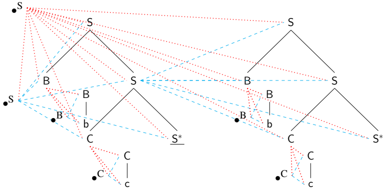
The 2d yield ...



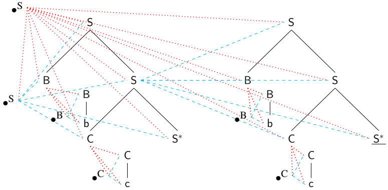
A non lexicalized 4d-TSG: a derivation

The 2d yield!

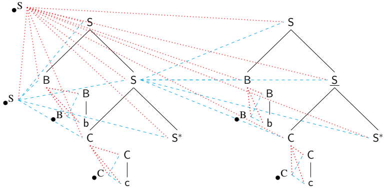




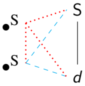
(w_2)



(w_3)

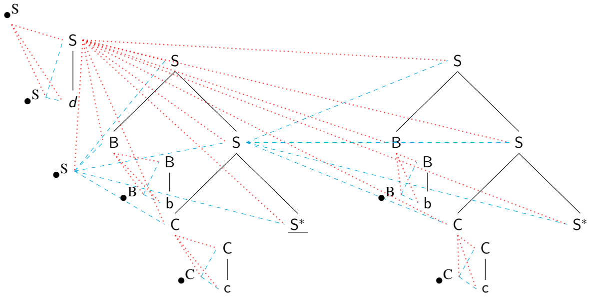


(w_1)

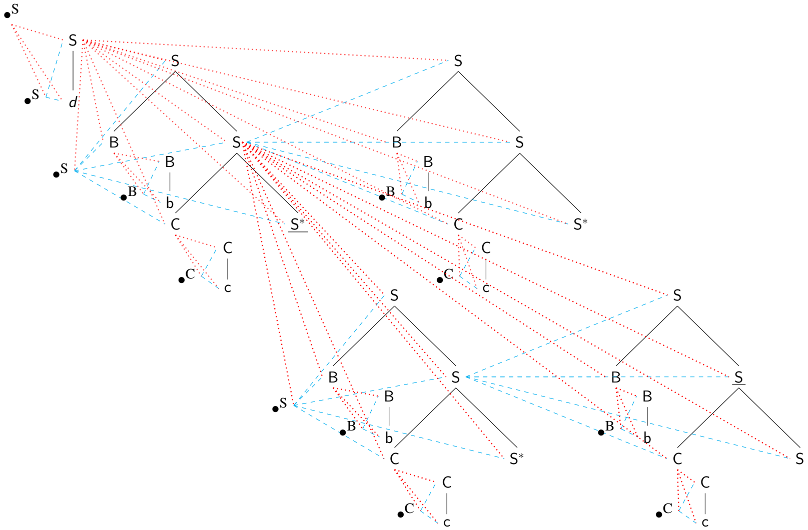


(i)

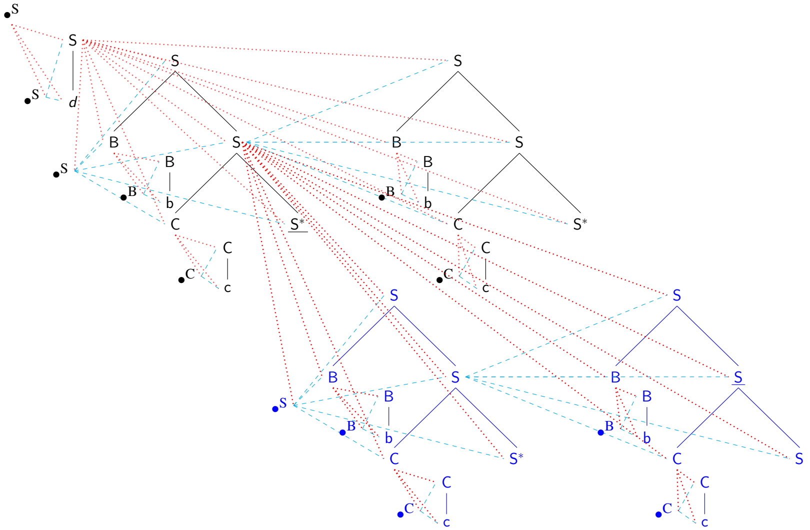
The lexicalized 4d-TAG: a derivation



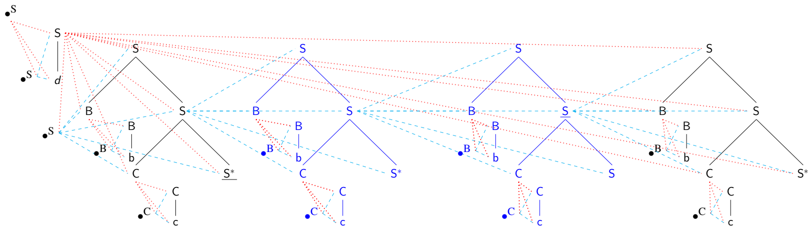
The lexicalized 4d-TAG: a derivation



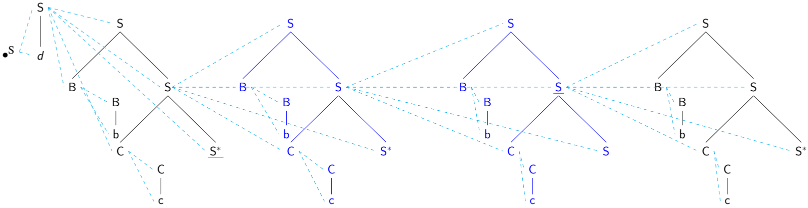
The lexicalized 4d-TAG: a derivation



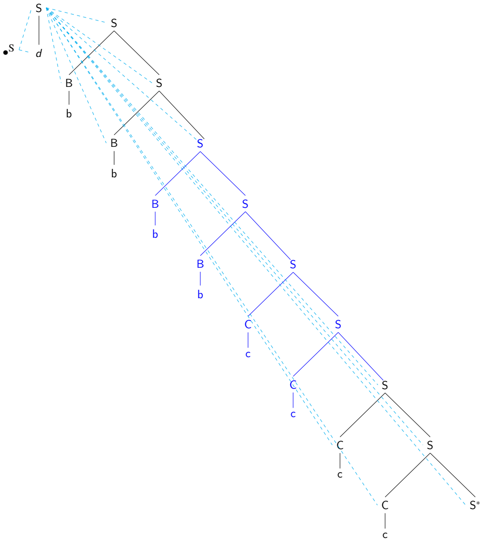
The 3d yield ...



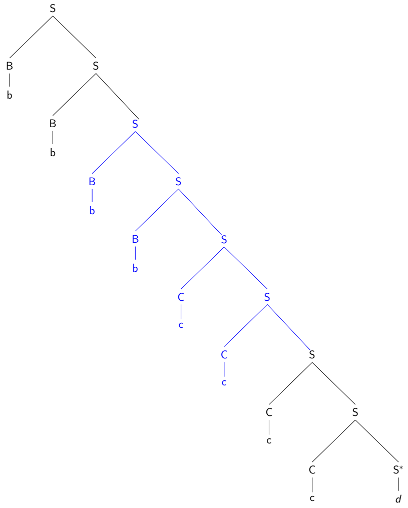
The 3d yield!



The 2d yield ...



The 2d yield!



Strong Lexicalization of d-TAGs

Proposition

For each finitely ambiguous d-dimensional TSG that does not generate the empty string and contains only useful trees, there is a strongly equivalent d-dimensional Lexicalized TAG.

but

d-TSGs are equivalent to (d - 1)-TAGs.

Proposition

(d+1)-TAGs strongly lexicalize d-TAGs

Strong Lexicalization of d-TAGs

Proposition

For each finitely ambiguous d-dimensional TSG that does not generate the empty string and contains only useful trees, there is a strongly equivalent d-dimensional Lexicalized TAG.

but

d-TSGs are equivalent to (d - 1)-TAGs.

Proposition

(d+1)-TAGs strongly lexicalize d-TAGs

Proposition

For each finitely ambiguous d-dimensional TSG that does not generate the empty string and contains only useful trees, there is a strongly equivalent d-dimensional Lexicalized TAG.

but

d-TSGs are equivalent to (d - 1)-TAGs.

Proposition

(d+1)-TAGs strongly lexicalize d-TAGs

Conclusion

- TAGs can be generalized to higher dimensional trees [Rogers, 2003]
- TAGs strongly lexicalize CFGs/TSGs [Schabes, 1990]

$\Rightarrow$ (d+1)-TAGs strongly lexicalize d-TAGs

TAGs as higher dimensional-trees

- lifting of existing results is straightforward
- increase in generative power
- what about parsing?

Conclusion

- TAGs can be generalized to higher dimensional trees [Rogers, 2003]
- TAGs strongly lexicalize CFGs/TSGs [Schabes, 1990]

$\Rightarrow$ (d+1)-TAGs strongly lexicalize d-TAGs

TAGs as higher dimensional-trees

- lifting of existing results is straightforward
- increase in generative power
- what about parsing?

Fujiyoshi, A. (2004).
Epsilon-free grammars and lexicalized grammars that generate the class of the mildly contextsensitive languages.
In *Proceedings of the 7th International Workshop on Tree Adjoining Grammar and Related Formalisms*, pages 16–23.

Kuhlmann, M. and Satta, G. (2012).
Tree-adjoining grammars are not closed under strong lexicalization.
Computational Linguistics, 38:617–629.

Maletti, A. and Engelfriet, J. (2012).
Strong lexicalization of tree adjoining grammars.
In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 506–515.

Selected References II

Rogers, J. (1998).
On defining TALs with logical constraints.
In Abeillé, A., Becker, T., Rambow, O., Satta, G., and Vijay-Shanker, K., editors, *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 151–154.

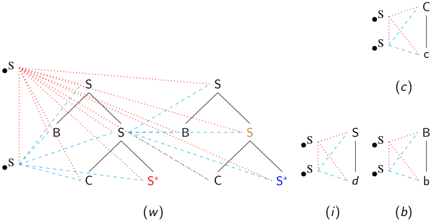
Rogers, J. (2003).
Syntactic structures as multi-dimensional trees.
Research on Language and Computation, 1:265–305.

Schabes, Y. (1990).
Mathematical and Computational Aspects of Lexicalized Grammars.
PhD thesis, Philadelphia, PA, USA.

Schabes, Y., Abeillé, A., and Joshi, A. K. (1988).
Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars.
Technical Report MS-CIS-88-65, Department of Computer & Information Science, University of Pennsylvania, Philadelphia, PA.

Lexicalization of a d-TSG: Step 1

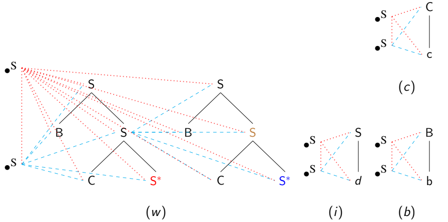
Step 1: Determine Recursion

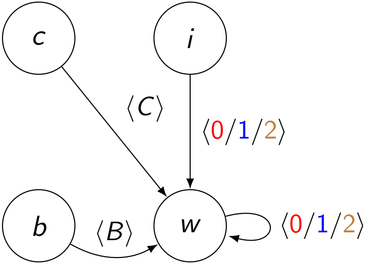


The TSG is Partitioned in Two Sets

$NR = \{b, c, i\}$
$R = \{I - NR\} = \{w\}$

Step 1: Determine Recursion



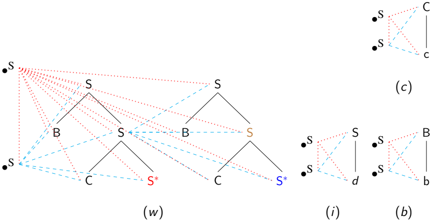


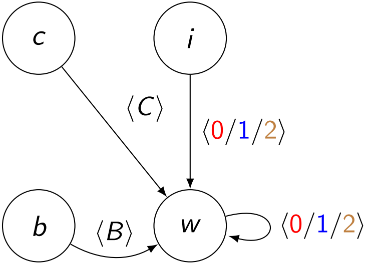
The TSG is Partitioned in Two Sets

$NR = \{b, c, i\}$

$R = \{I - NR\} = \{w\}$

Step 1: Determine Recursion





The TSG is Partitioned in Two Sets

$NR = \{b, c, i\}$

$R = \{I - NR\} = \{w\}$

Step 2: Determine the set I_{lex}.

- $T(NR)$: the closure of NR under adjunction

I_{lex}

is the maximal subset of $T(NR)$ that only contains d-trees whose root is labeled by the start category S.

$$I_{lex} = \{i\}$$

Lexicalization of a d-TSG: Step 2

Step 2: Determine the set I_{lex}.

- $T(NR)$: the closure of NR under adjunction

I_{lex}

is the maximal subset of $T(NR)$ that only contains d-trees whose root is labeled by the start category S.

$$I_{lex} = \{i\}$$

Lexicalization of a d-TSG: Step 2

Step 2: Determine the set I_{lex}.

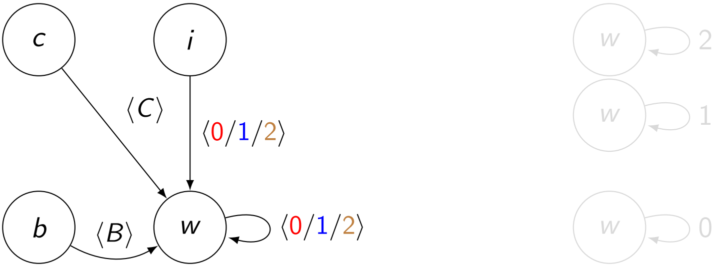
- $T(NR)$: the closure of NR under adjunction

I_{lex}

is the maximal subset of $T(NR)$ that only contains d-trees whose root is labeled by the start category S.

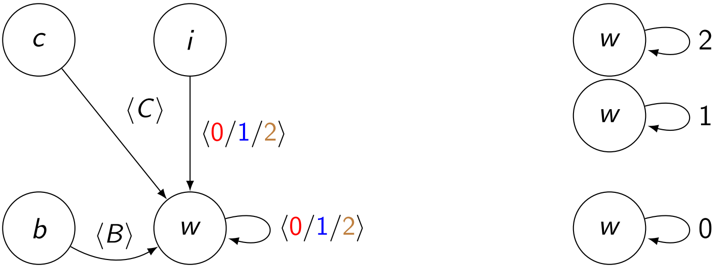
$$I_{lex} = \{i\}$$

Step 3: Compute Base Cycles



Lexicalization of a d-TSG: Step 2

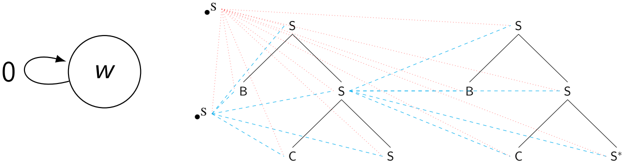
Step 3: Compute Base Cycles

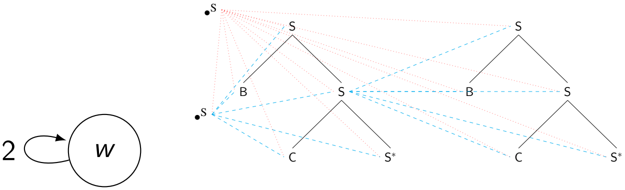


Lexicalization of a d-TSG: Step 4

Step 4: Determine A_{lex}

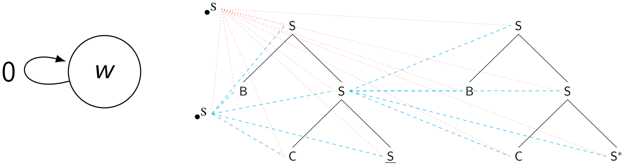
- expand base cycles;
- relabel $3d$ foot node;
- exhaustive substitution;

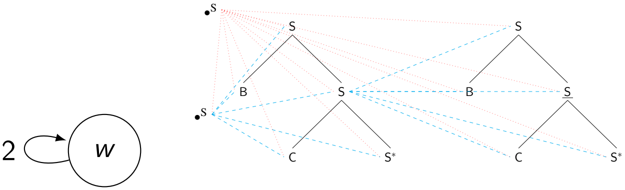




Step 4: Determine A_{lex}

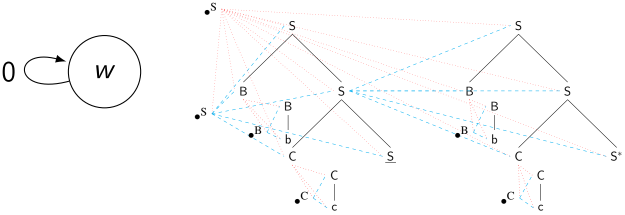
- expand base cycles;
- relabel $3d$ foot node;
- exhaustive substitution;

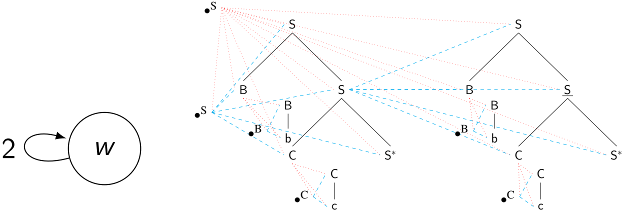




Step 4:
Determine A_{lex}

- expand base cycles;
- relabel $3d$ foot node;
- exhaustive substitution;



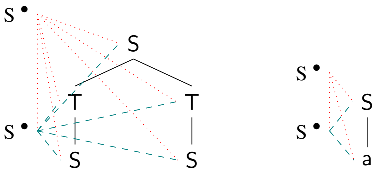


One final question

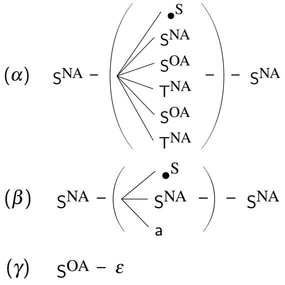
Are *d*-dimensional TAGs closed under strong lexicalization?

[Kuhlmann and Satta, 2012]

TAGs are not closed under strong lexicalization



(α) (β)



Excess

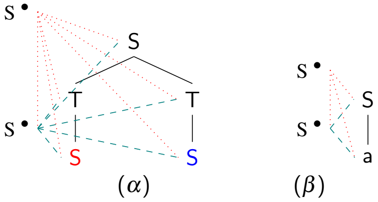
measures the distance between a root node and a terminal node

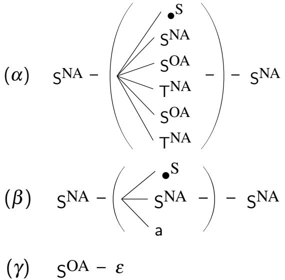
One final question

Are *d*-dimensional TAGs closed under strong lexicalization?

[Kuhlmann and Satta, 2012]

TAGs are not closed under strong lexicalization



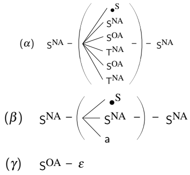


Excess

measures the distance between a root node and a terminal node

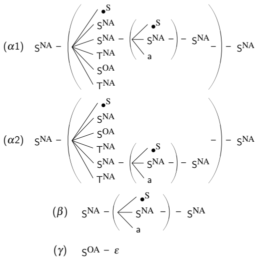
d-TAGs are not closed under strong lexicalization

Non Lexicalized



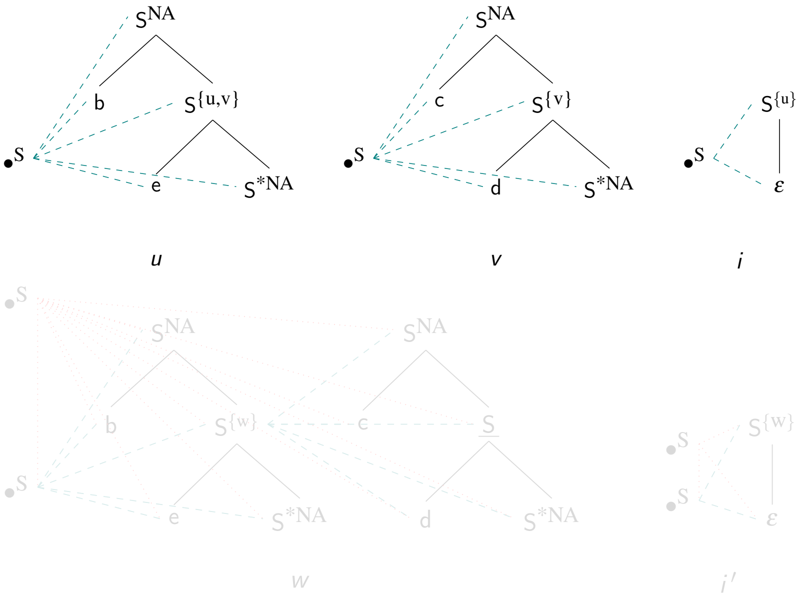
max. excess of node *a* is unbounded

Lexicalized

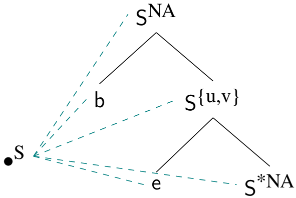


max excess of node *a* is 2

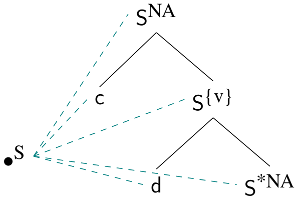
Increasing the power by increasing the dimensionality: Ex. 2



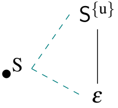
Increasing the power by increasing the dimensionality: Ex. 2



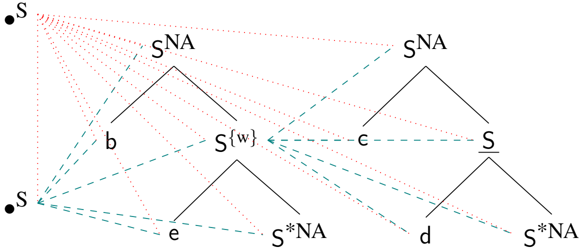
u



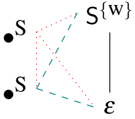
v



i



w



i′

Map of Existing Results