Unsupervised Generation of Parallel Treebanks through Sub-Tree Alignment

Ventsislav Zhechev
contact@ventsislavzhechev.eu

Talk Outline

- What is a Parallel Treebank and Why do we Need One?
- System Design and Features
- Software Package Details
- Avenues for Improvement
- Conclusions

What is a Parallel Treebank?

What is a Parallel Treebank?

English

I do not think it is necessary for classic cars to be part of the directive .

I am not looking for such rigidly high recycling quotas when it comes to special-purpose vehicles either .

I want special-purpose vehicles such as ambulances to have high recovery quotas .

This is my main concern in this matter .

German

Ich halte es nicht für notwendig , daß Oldtimer Bestandteil dieser Richtlinie sind .

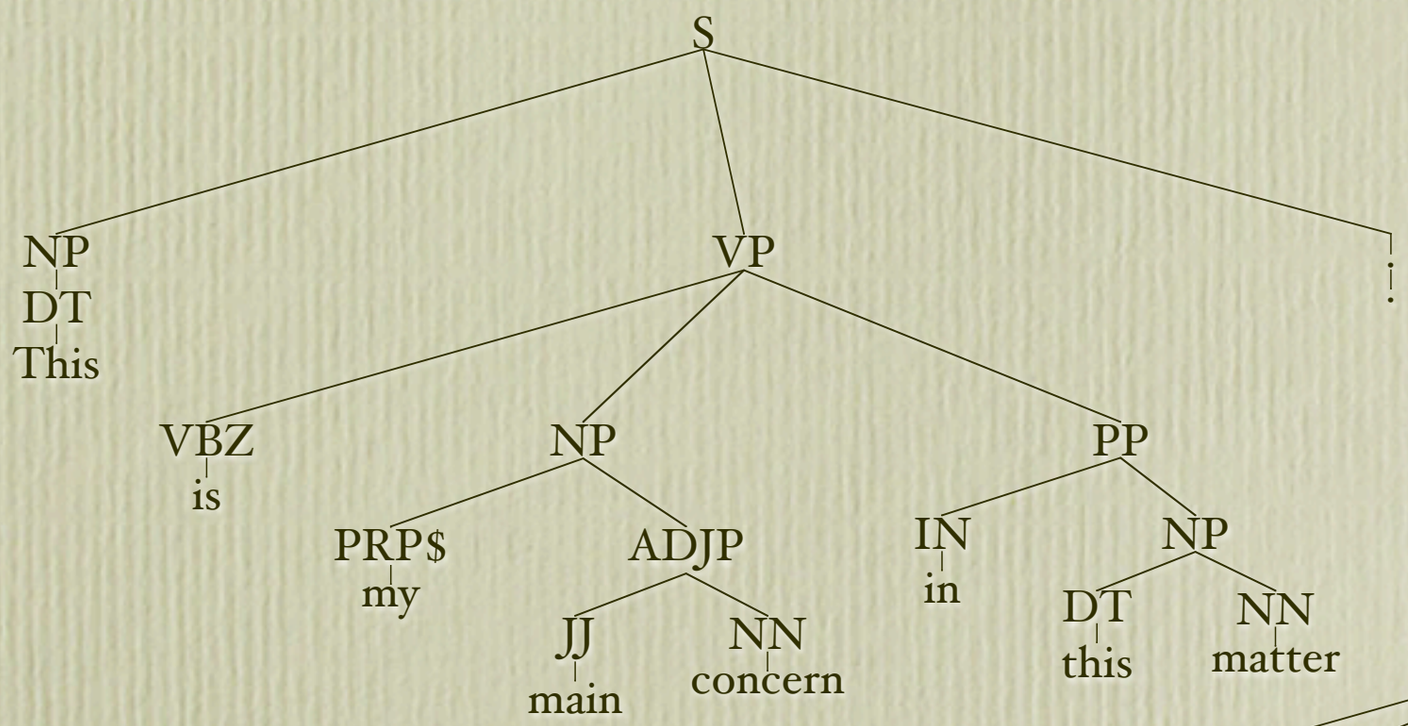
Auch bei Sonderfahrzeugen strebe ich nicht so unbedingt hohe Recyclingquoten an .

Ich habe den Wunsch , daß Sonderfahrzeuge wie Krankenwagen hohe Rettungsquoten haben .

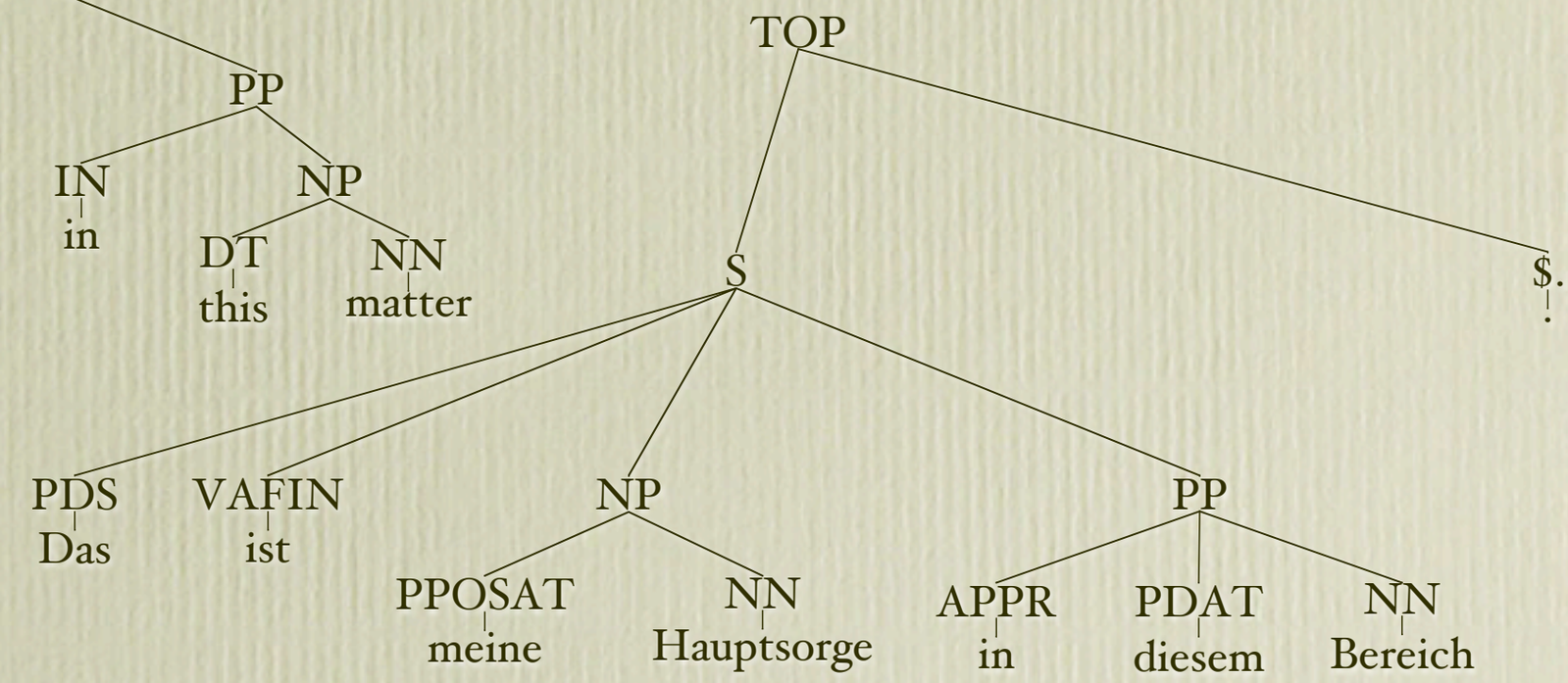
Das ist meine Hauptsorge in diesem Bereich .

What is a Parallel Treebank?

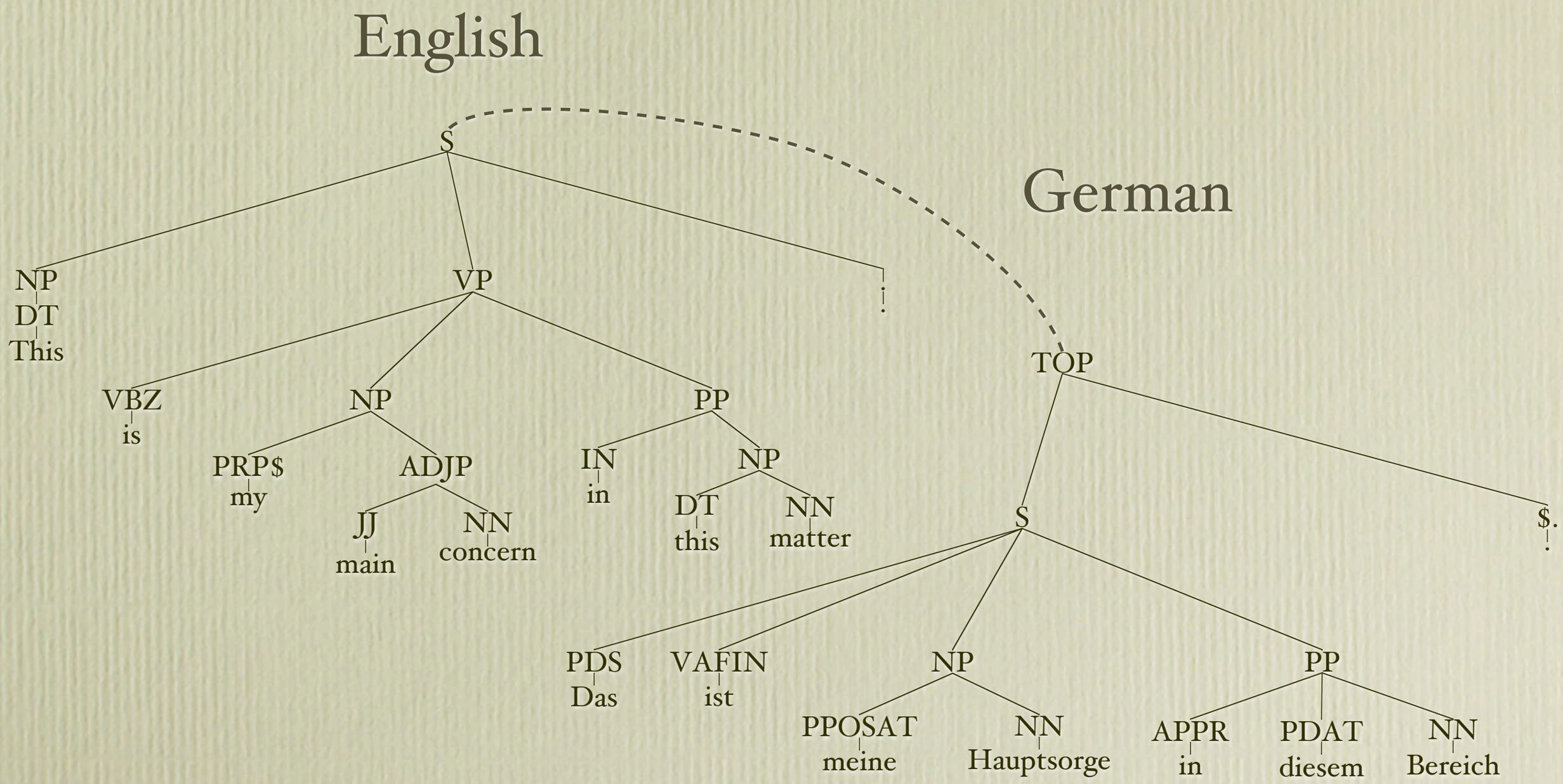
English



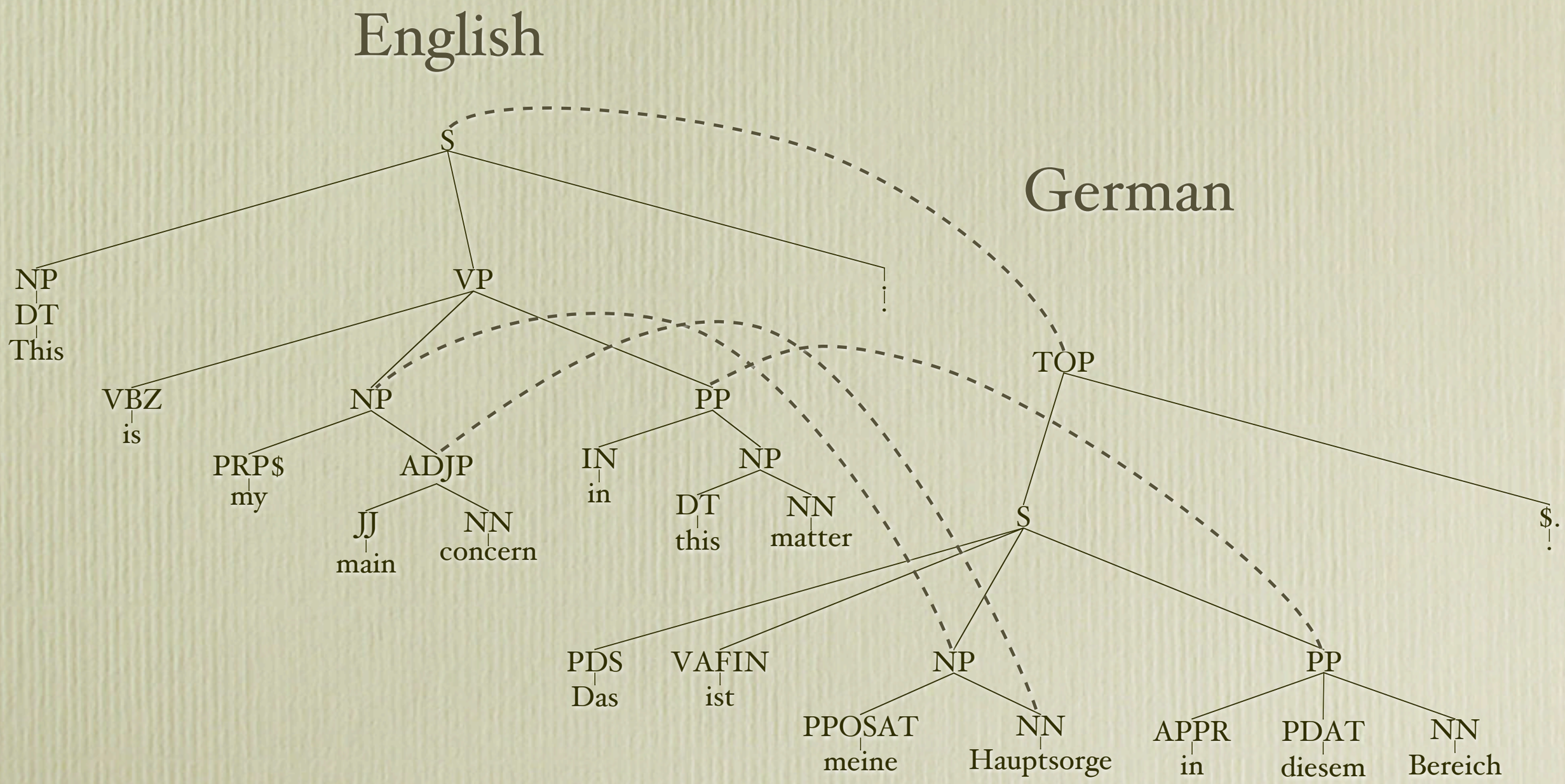
German



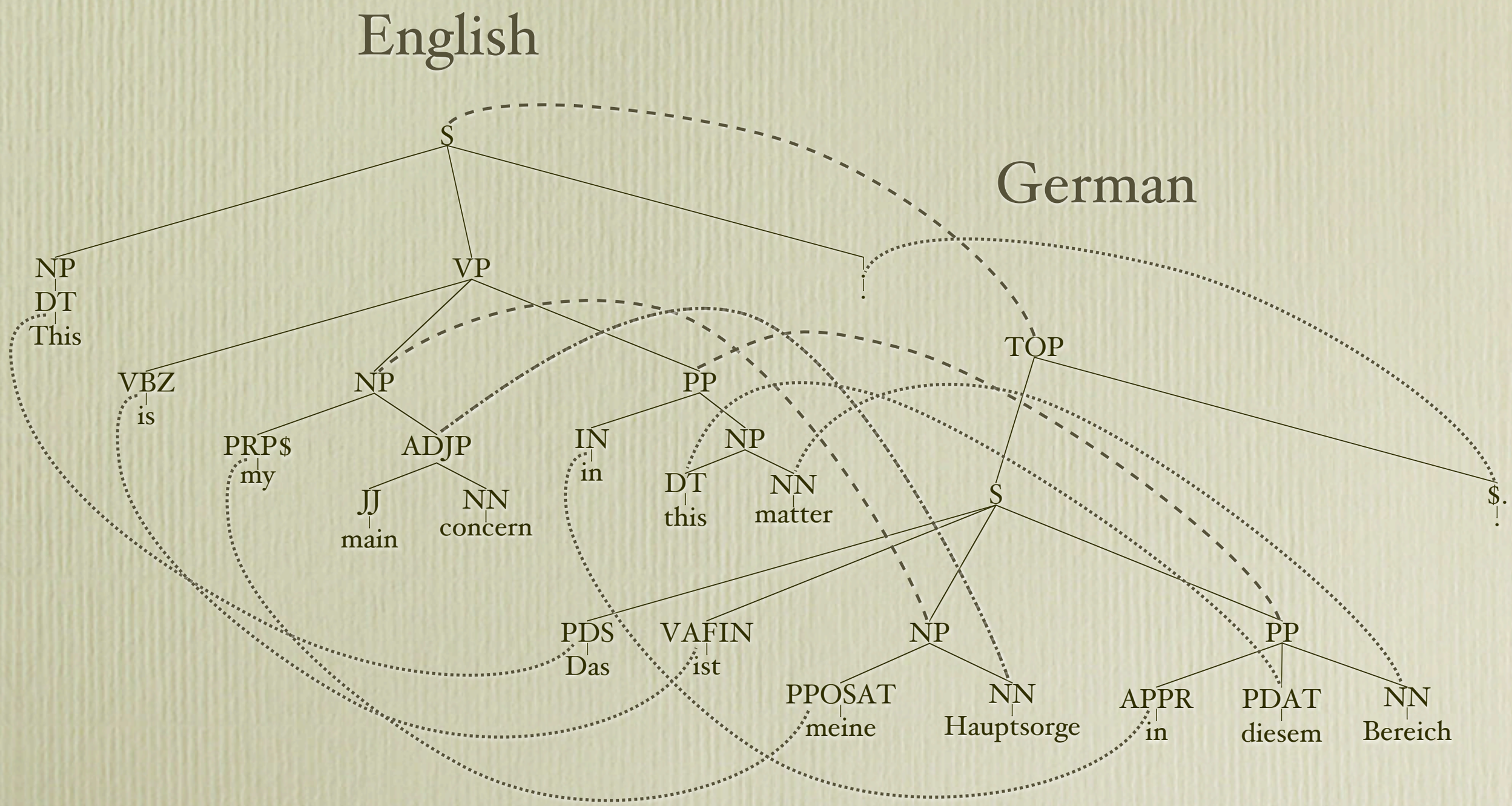
What is a Parallel Treebank?



What is a Parallel Treebank?



What is a Parallel Treebank?

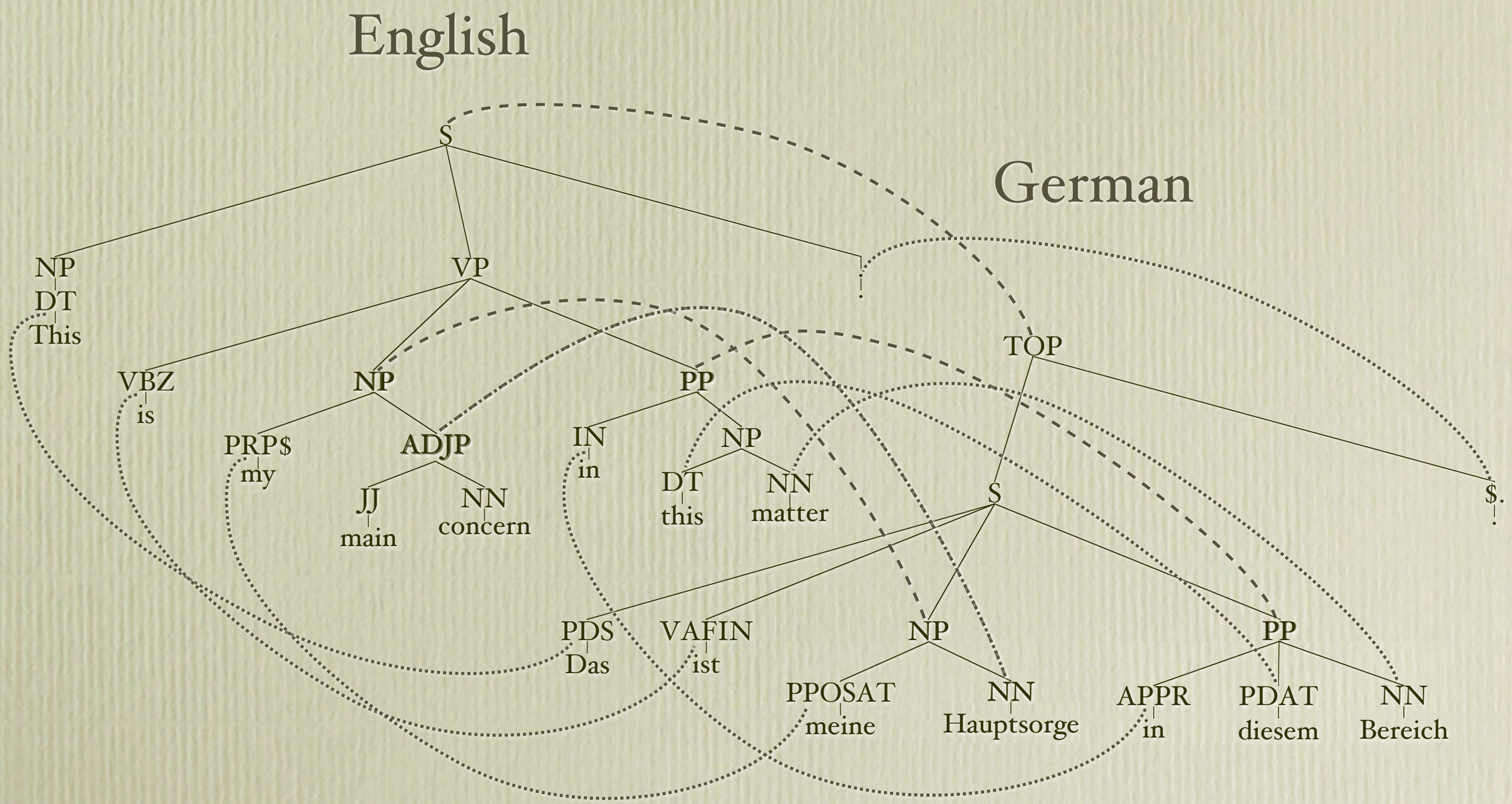


Uses of Parallel Treebanks

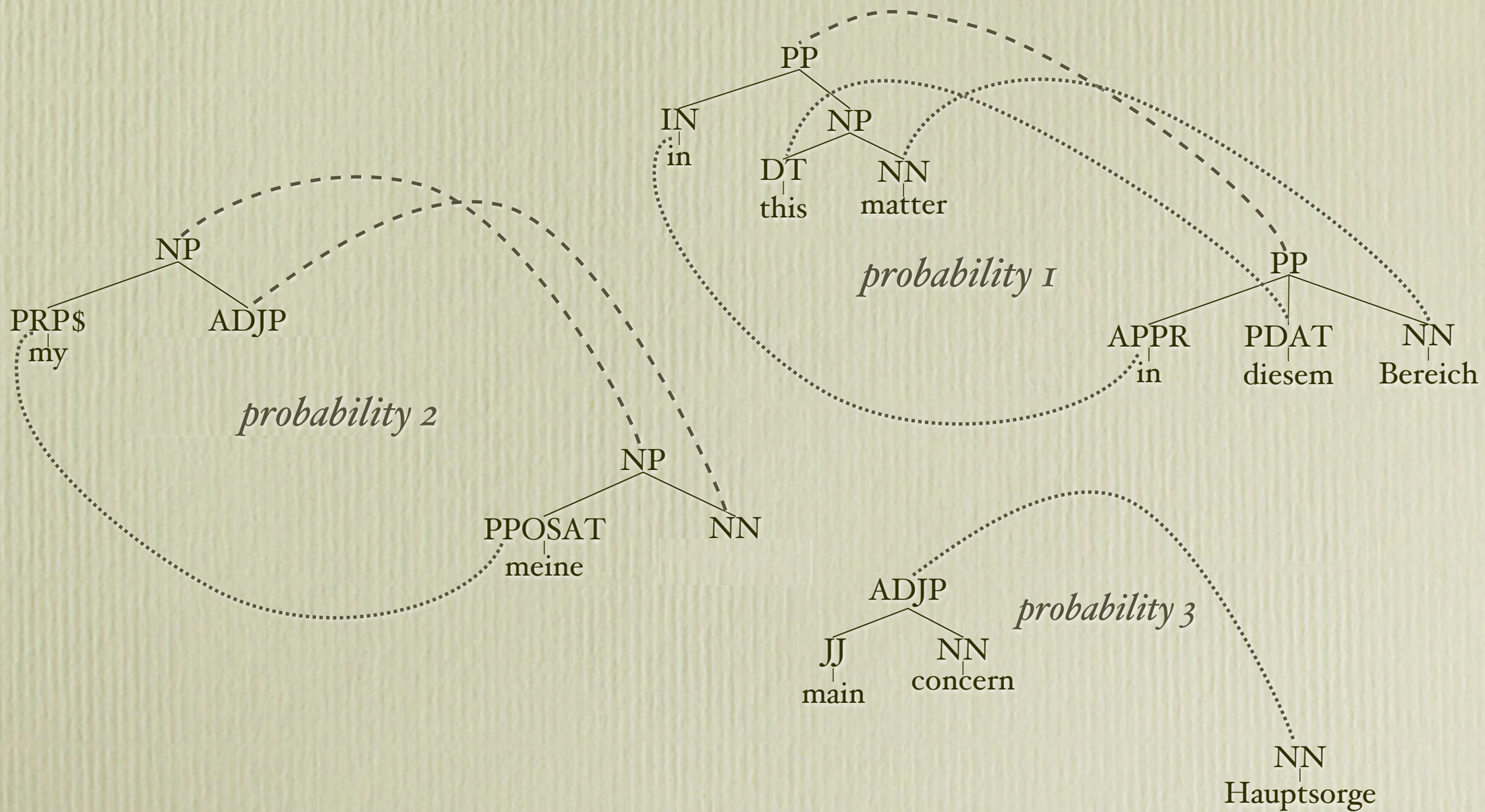
Uses of Parallel Treebanks

- Hiero (Chiang, 2007)
- Probabilistic Synchronous Tree-Insertion Grammars (Nesson et al., 2006)
- Data-Oriented Translation (Hearne and Way, 2006)
- Stat-XFER (Lavie, 2008)

Uses of Parallel Treebanks



Uses of Parallel Treebanks



What do we need to generate Parallel Treebanks?

What do we need to generate Parallel Treebanks?

- We already have:
  - Parallel Corpora
  - Parsers
- What's missing?
  - A Sub-Tree Aligner
    - for existing research see
      (Tinsley et al., 2007), MT Summit XI
      (Zhechev, to appear), PhD Thesis

What do we want in a Sub-Tree Aligner?

What do we want in a Sub-Tree Aligner?

- Independence
- Preservation
- Minimal External Resources
- Guided Lexical Alignments

How does the Sub-Tree Aligner Work?

How does the Sub-Tree Aligner Work?

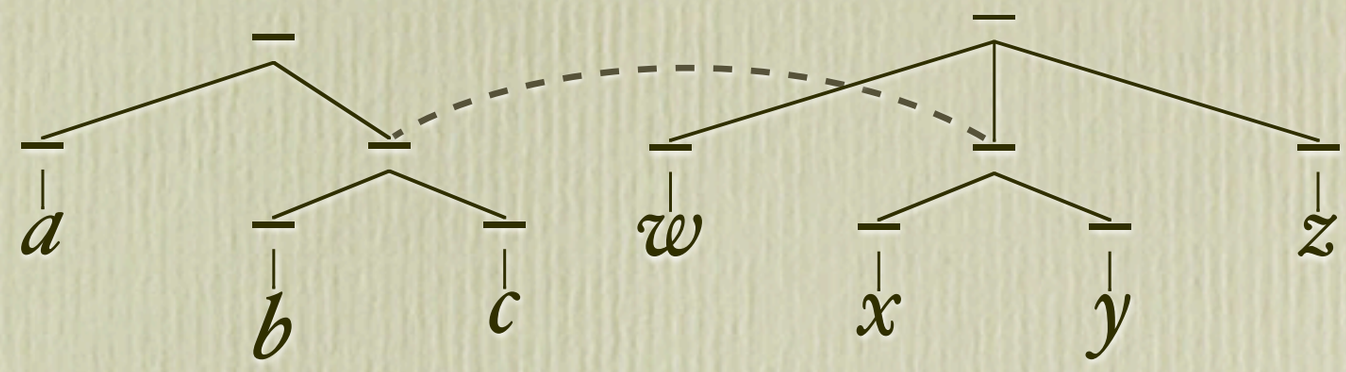
- Prerequisites:
  - sentence-aligned parallel text
  - monolingual parsers for both languages
    - source-to-target and target-to-source word-alignment probability tables
- Both sides of the parallel text need to be parsed in advance
- The sub-tree aligner operates on one parsed sentence pair at a time

How does the Sub-Tree Aligner Work?

- Initialisation
  - extract the relevant word-alignment data
  - calculate scores for all possible links between nodes in the source and target tree
    - only nonzero scores are stored
- Selection
  - select the best combination of links for the sentence pair
  - two selection algorithms

Translational Equivalence

Translational Equivalence



inside

$s_l = \langle b\ c \rangle$

$t_l = \langle x\ y \rangle$

outside

$\bar{s}_l = \langle a \rangle$

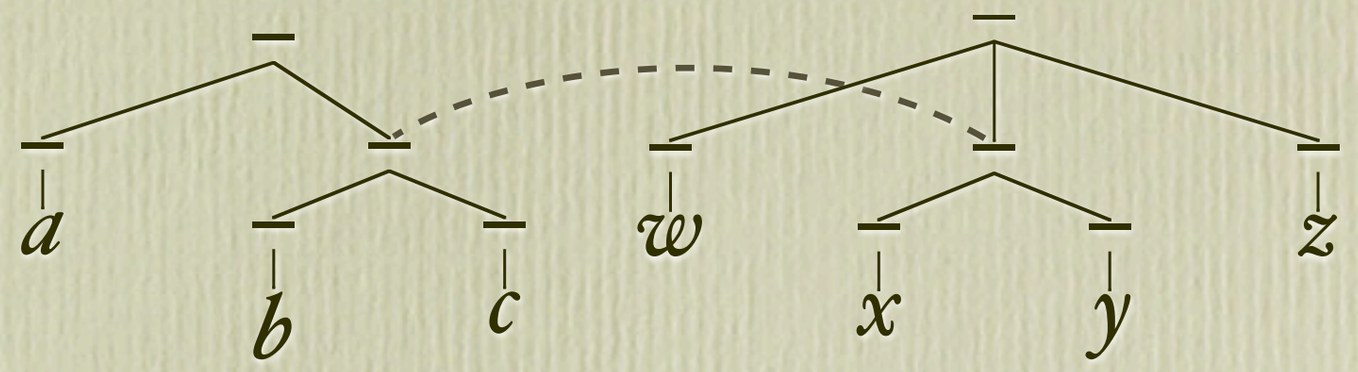
$\bar{t}_l = \langle w\ z \rangle$

$$\gamma(\langle s, t \rangle) = \alpha(s_l|t_l) \cdot \alpha(t_l|s_l) \cdot \alpha(\bar{s}_l|\bar{t}_l) \cdot \alpha(\bar{t}_l|\bar{s}_l)$$

score1

$$\alpha(y|x) = \prod_{i}^{|x|} \sum_{j}^{|y|} P(y_j|x_i)$$

Translational Equivalence



inside

$s_l = \langle b\ c \rangle$

$t_l = \langle x\ y \rangle$

outside

$\bar{s}_l = \langle a \rangle$

$\bar{t}_l = \langle w\ z \rangle$

$$\gamma(\langle s, t \rangle) = \alpha(s_l|t_l) \cdot \alpha(t_l|s_l) \cdot \alpha(\bar{s}_l|\bar{t}_l) \cdot \alpha(\bar{t}_l|\bar{s}_l)$$

score2

$$\alpha(y|x) = \prod_{j}^{|y|} \sum_{i}^{|x|} P(y_j|x_i) \Big/ |x|$$

Greedy-Search Selection

Greedy-Search Selection

```
while unprocessed hypotheses remain do
 link the highest-scoring hypothesis
 discard all incompatible hypotheses
end while
```

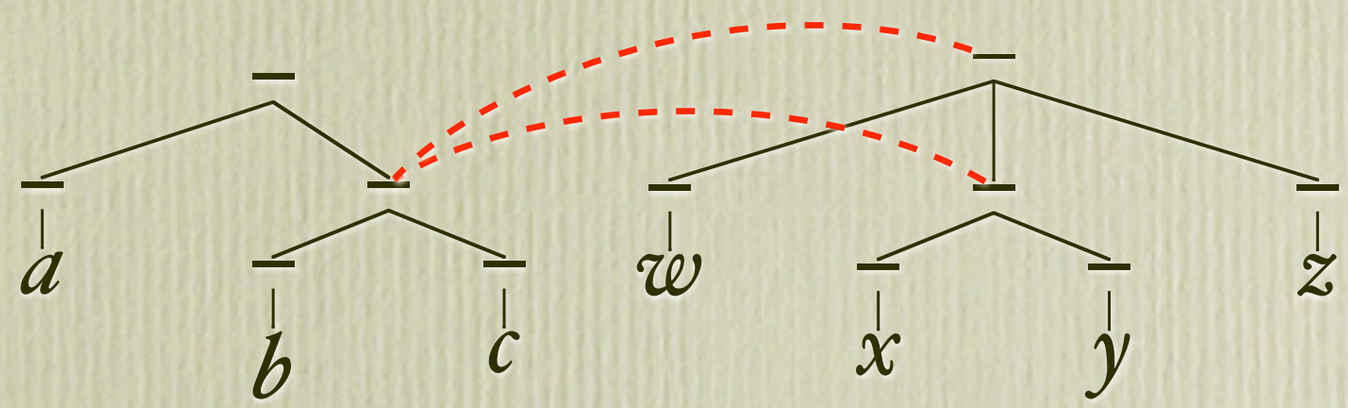
Greedy-Search Selection

```
while unprocessed hypotheses remain do
 link the highest-scoring hypothesis
 discard all incompatible hypotheses
end while
```

- Two hypotheses are incompatible if
  - they share a node

Greedy-Search Selection

```
while unprocessed hypotheses remain do
 link the highest-scoring hypothesis
 discard all incompatible hypotheses
end while
```



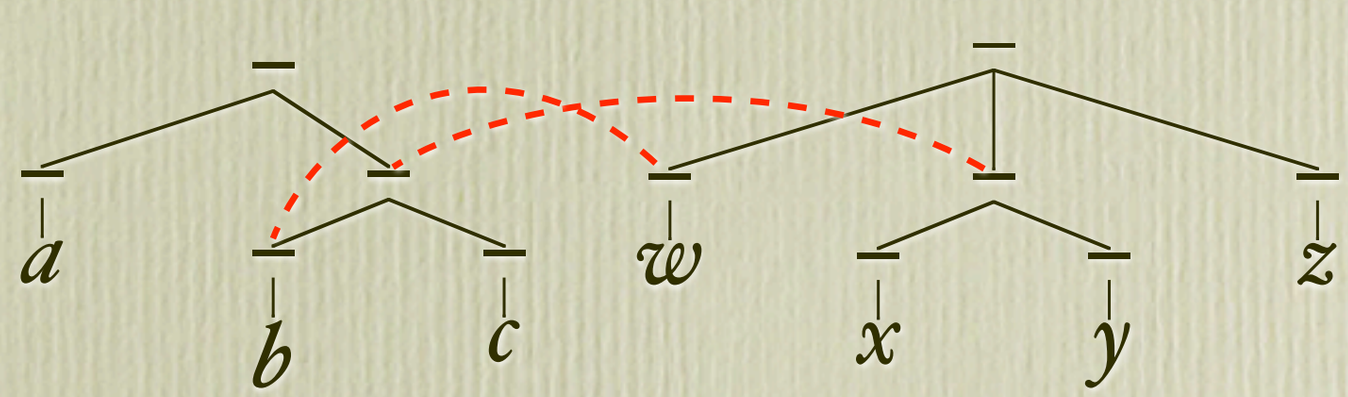
Greedy-Search Selection

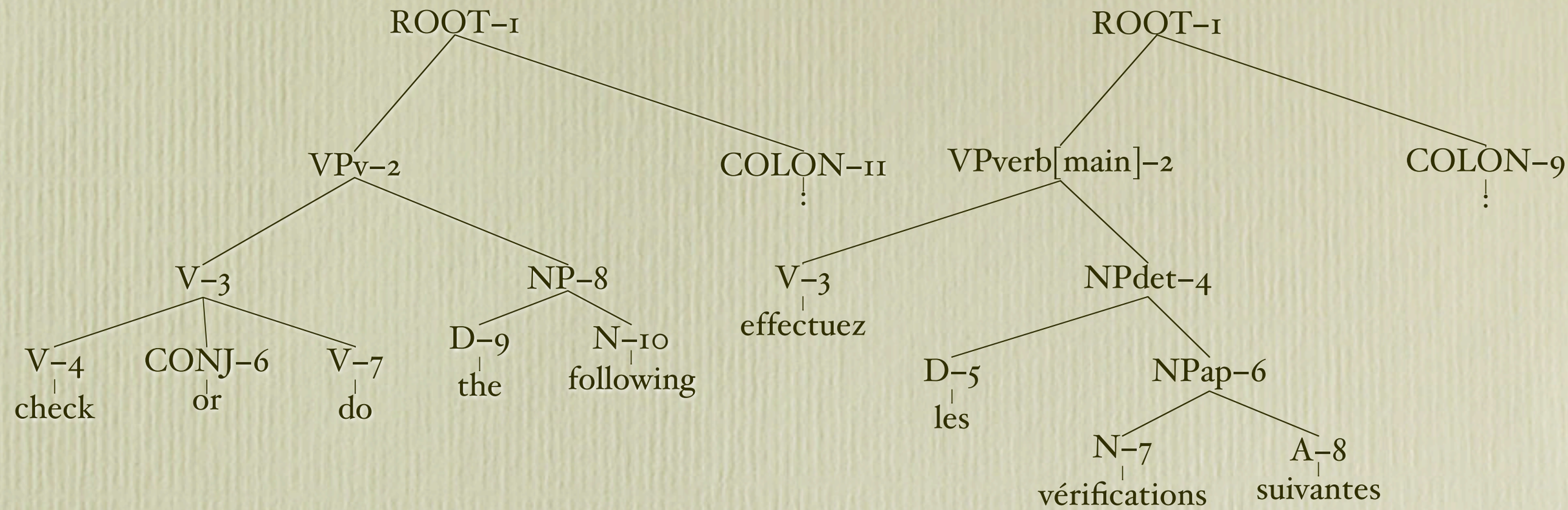
```
while unprocessed hypotheses remain do
 link the highest-scoring hypothesis
 discard all incompatible hypotheses
end while
```

- Two hypotheses are incompatible if
  - they share a node
  - descendants / ancestors of the source node are linked to non-descendants / non-ancestors of the target node

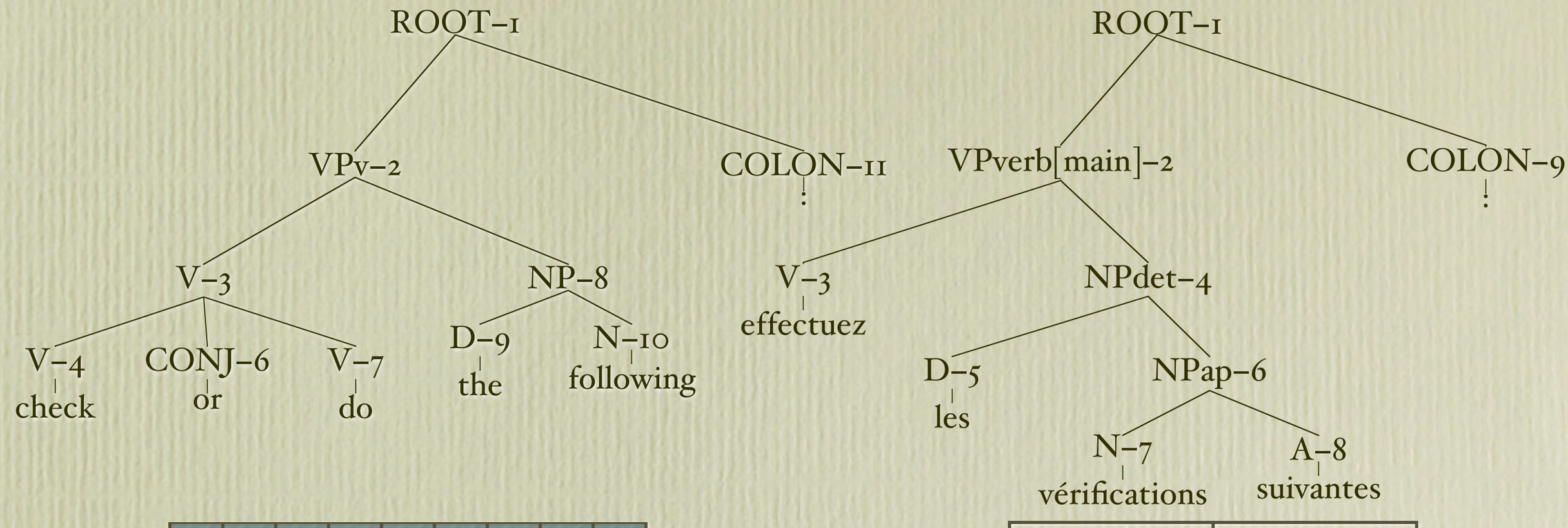
Greedy-Search Selection

```
while unprocessed hypotheses remain do
 link the highest-scoring hypothesis
 discard all incompatible hypotheses
end while
```





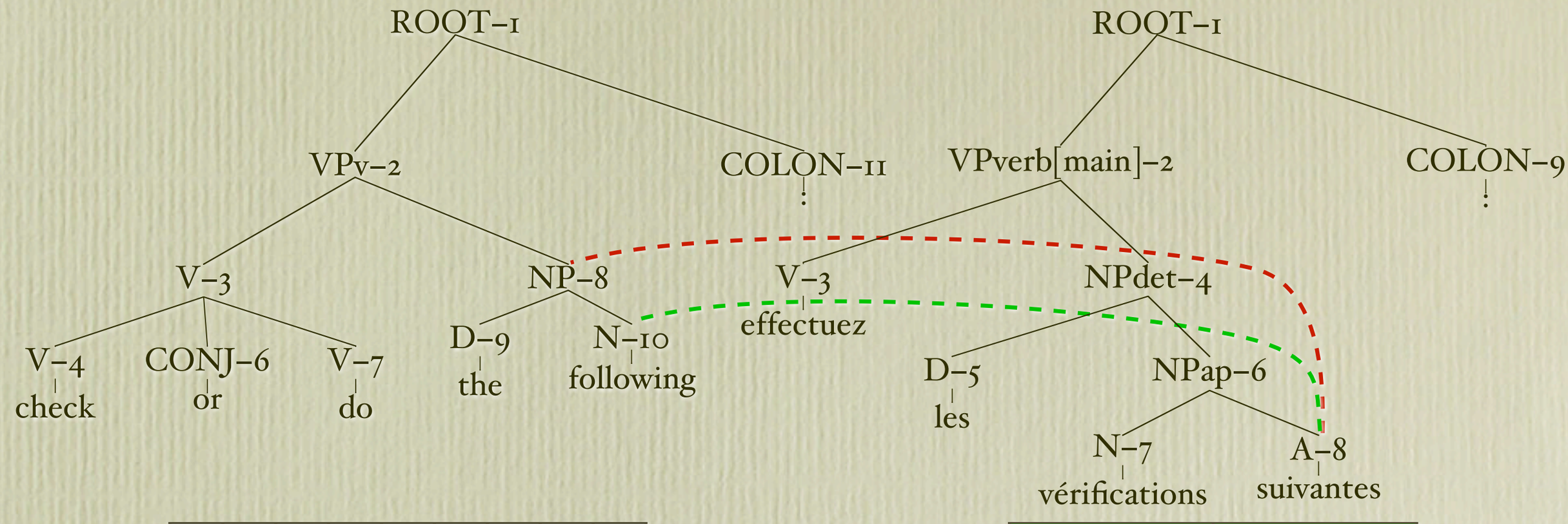
	1	2	3	4	5	6	7	8	9
1									
2									
3									
4									
6									
7									
8									
9									
10									
11									



	1	2	3	4	5	6	7	8	9
1			■	■	■	■	■	■	■
2			■	■	■	■	■	■	■
3	■	■	■	■	■	■	■	■	■
4	■	■		■	■	■	■	■	■
6	■	■	■	■	■	■	■	■	■
7	■	■	■	■	■	■	■	■	■
8	■	■	■	■	■	■	■		■
9	■	■	■	■		■	■	■	
10	■	■	■	■	■	■	■		■
11	■	■	■	■	■	■	■	■	

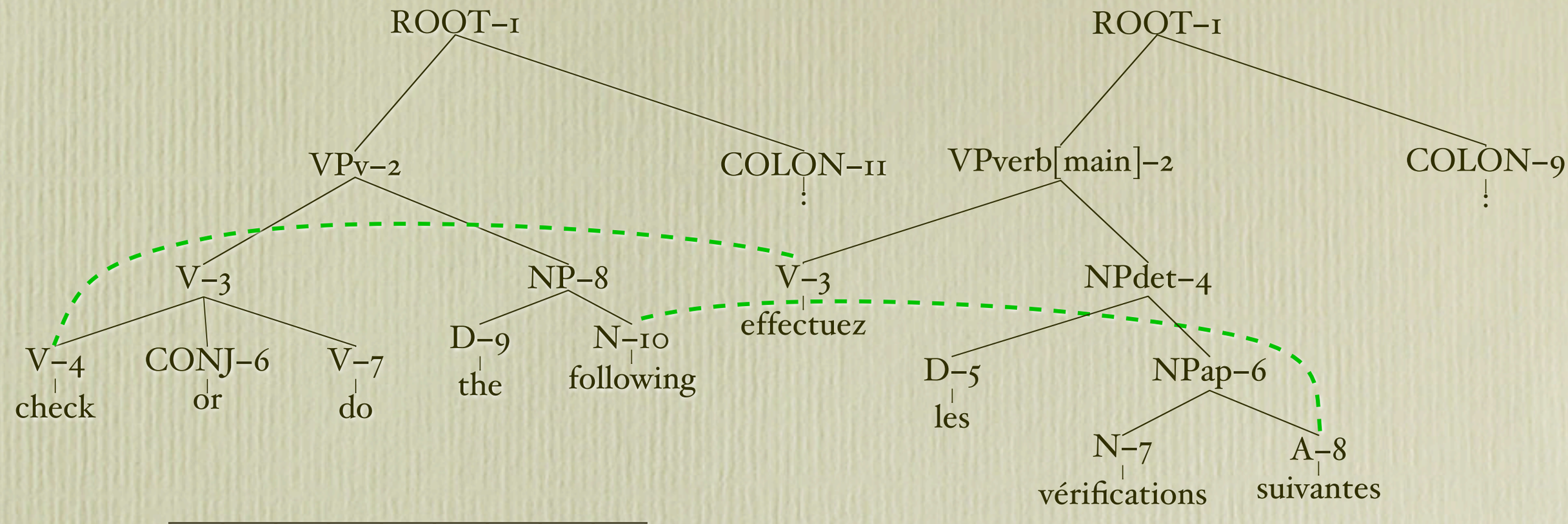
10 ⇔ 8	score_1
4 ⇔ 3	score_2
1 ⇔ 1	score_3
8 ⇔ 8	score_4
11 ⇔ 9	score_5
2 ⇔ 2	score_6
9 ⇔ 5	score_7
1 ⇔ 2	score_8
2 ⇔ 1	score_9
9 ⇔ 9	score_10

1.e-23 ÷ 5.e-17



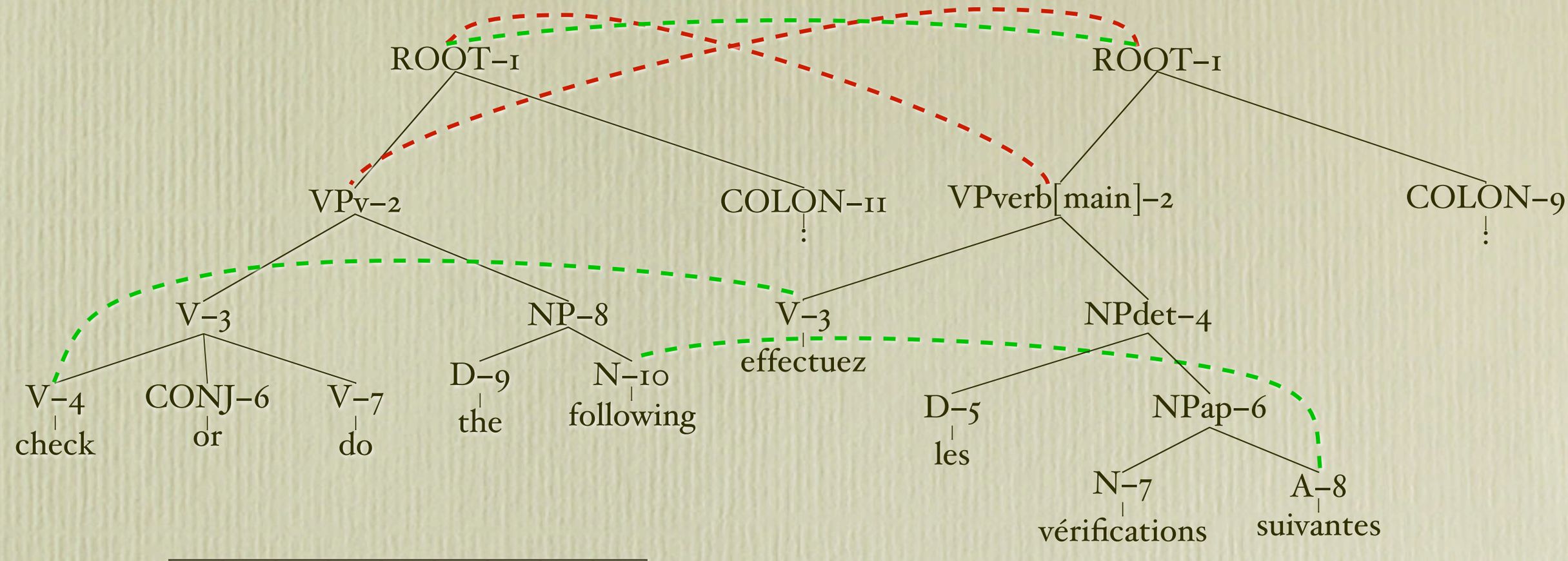
	1	2	3	4	5	6	7	8	9
1									
2									
3									
4									
6									
7									
8								1	
9									
10								1	
11									

10 ⇔ 8	score_1
4 ⇔ 3	score_2
1 ⇔ 1	score_3
8 ⇔ 8	score_4
11 ⇔ 9	score_5
2 ⇔ 2	score_6
9 ⇔ 5	score_7
1 ⇔ 2	score_8
2 ⇔ 1	score_9
9 ⇔ 9	score_10



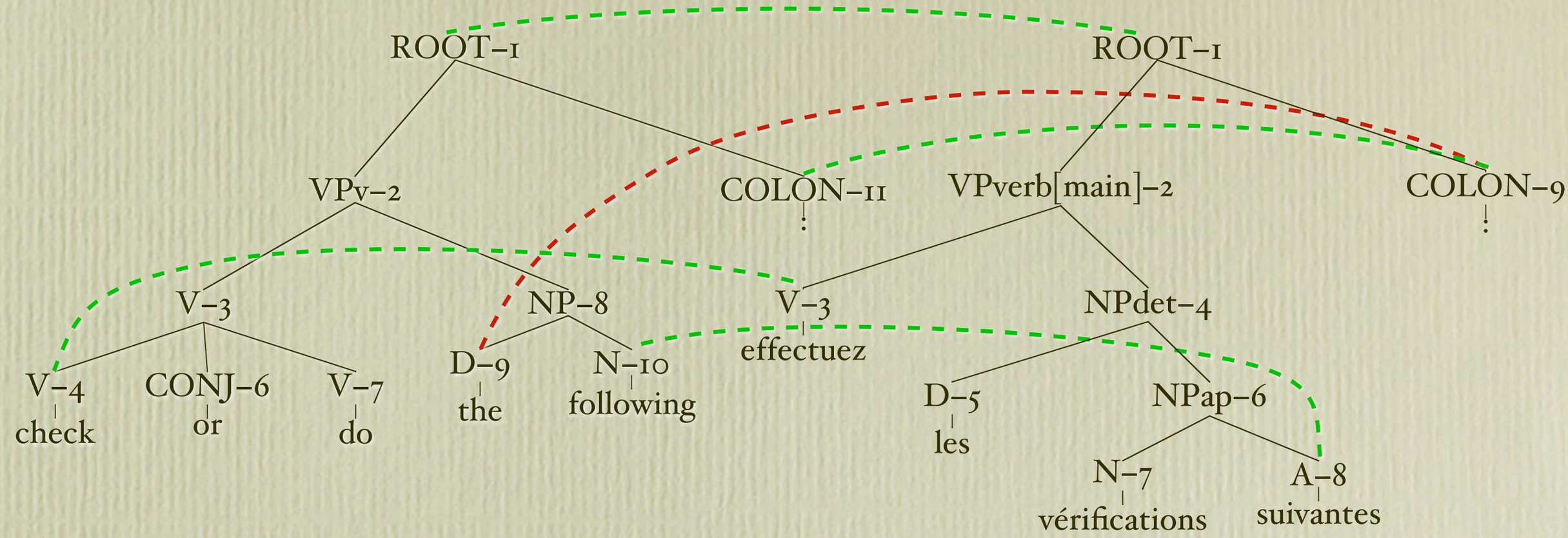
	1	2	3	4	5	6	7	8	9
1									
2									
3									
4			2						
6									
7									
8								1	
9									
10								1	
11									

4 ⇔ 3	score_2
1 ⇔ 1	score_3
11 ⇔ 9	score_5
2 ⇔ 2	score_6
9 ⇔ 5	score_7
1 ⇔ 2	score_8
2 ⇔ 1	score_9
9 ⇔ 9	score_10



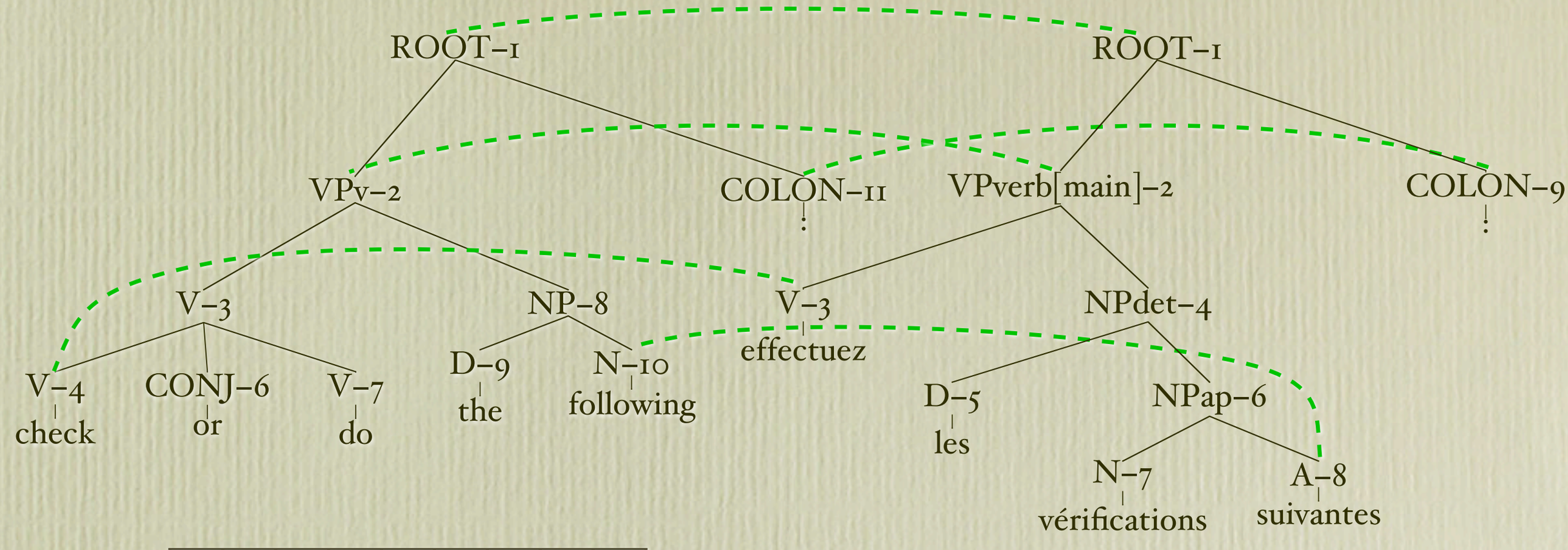
	1	2	3	4	5	6	7	8	9
1	3	3							
2	3								
3									
4			2						
6									
7									
8								1	
9									
10								1	
11									

1 ⇔ 1	score_3
11 ⇔ 9	score_5
2 ⇔ 2	score_6
9 ⇔ 5	score_7
1 ⇔ 2	score_8
2 ⇔ 1	score_9
9 ⇔ 9	score_10



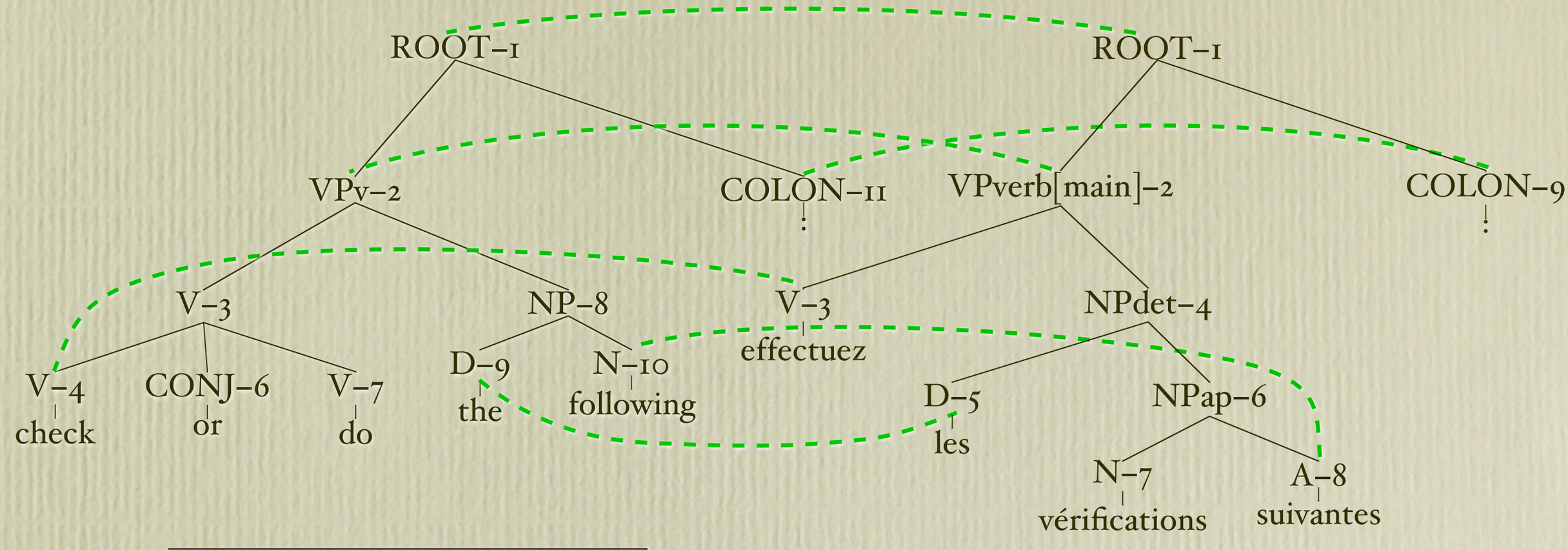
	1	2	3	4	5	6	7	8	9
1	3	3							
2	3								
3									
4			2						
6									
7									
8								1	
9									4
10								1	
11									4

11 ⇔ 9	score_5
2 ⇔ 2	score_6
9 ⇔ 5	score_7
9 ⇔ 9	score_10



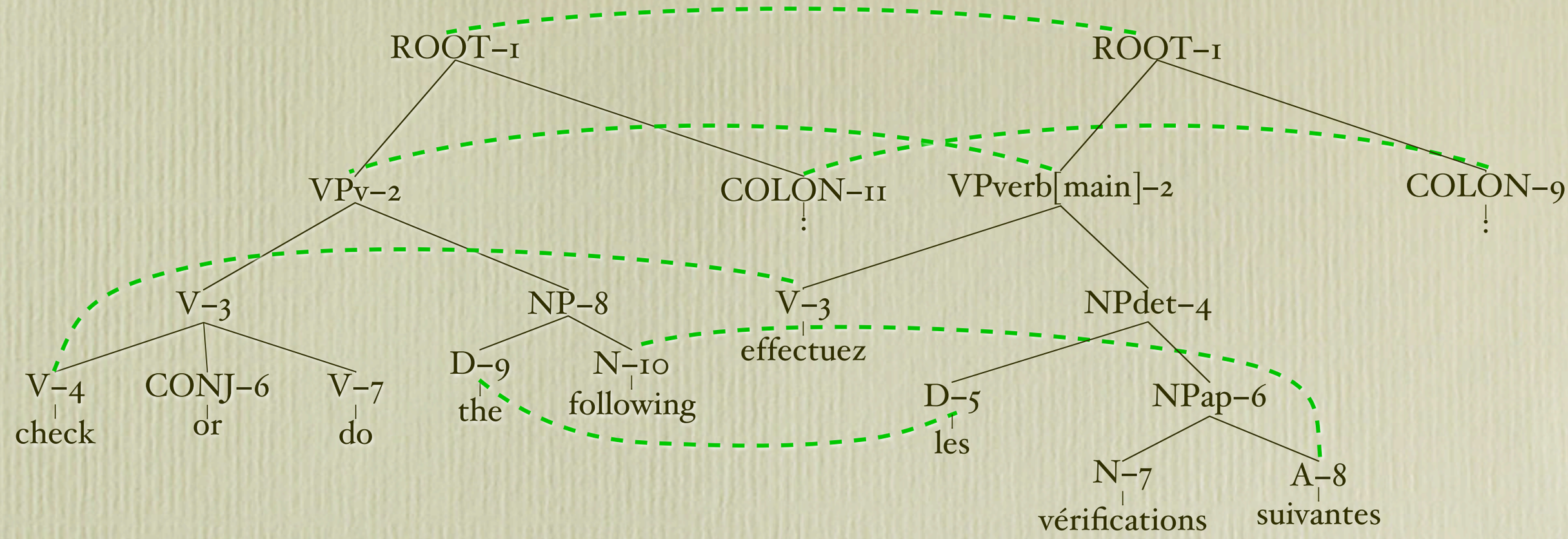
	1	2	3	4	5	6	7	8	9
1	3	3							
2	3	5							
3									
4			2						
6									
7									
8								1	
9									4
10								1	
11									4

2 ⇔ 2	score_6
9 ⇔ 5	score_7



	1	2	3	4	5	6	7	8	9
1	3	3							
2	3	5							
3									
4			2						
6									
7									
8								1	
9					6				4
10								1	
11									4

9 ⇔ 5	score_7



	1	2	3	4	5	6	7	8	9
1	3	3							
2	3	5							
3									
4			2						
6									
7									
8								1	
9					6				4
10								1	
11									4

Greedy-Search Selection

```
while unprocessed hypotheses remain do
 link the highest-scoring hypothesis
 discard all incompatible hypotheses
end while
```

- Two hypotheses are incompatible if
  - they share a node
  - descendants / ancestors of the source node are linked to non-descendants / non-ancestors of the target node
- Several hypotheses may share the highest translational equivalence score!

skip1

```
while unprocessed hypotheses with no tied competitors remain do
 while the highest-scoring hypothesis has tied competitors do
  skip all tied competitors
 end while
 link the highest-scoring hypothesis
 discard all incompatible hypotheses
end while
```

a ⇔ b	score_1
a ⇔ c	score_1
d ⇔ c	score_2
e ⇔ f	score_3
g ⇔ b	score_4
h ⇔ i	score_4

skip1

```
while unprocessed hypotheses with no tied competitors remain do
 while the highest-scoring hypothesis has tied competitors do
  skip all tied competitors
 end while
 link the highest-scoring hypothesis
 discard all incompatible hypotheses
end while
```

a ⇔ b	score_1
a ⇔ c	score_1
d ⇔ c	score_2
e ⇔ f	score_3
g ⇔ b	score_4
h ⇔ i	score_4

skip1

```
while unprocessed hypotheses with no tied competitors remain do
 while the highest-scoring hypothesis has tied competitors do
  skip all tied competitors
 end while
 link the highest-scoring hypothesis
 discard all incompatible hypotheses
end while
```

a ⇔ b	score_1
e ⇔ f	score_3
g ⇔ b	score_4
h ⇔ i	score_4

skip2

```
while unprocessed hypotheses with no tied competitors remain do
 if the highest-scoring hypothesis has tied competitors do
  mark the constituents of all tied competitors
 end if
 while the highest-scoring hypothesis has a marked constituent do
  skip
 end while
 link the highest-scoring hypothesis
 discard all incompatible hypotheses
 unmark all constituents
end while
```

a ⇔ b	score_1
a ⇔ c	score_1
d ⇔ c	score_2
e ⇔ f	score_3
g ⇔ b	score_4
h ⇔ i	score_4

skip2

```
while unprocessed hypotheses with no tied competitors remain do
 if the highest-scoring hypothesis has tied competitors do
  mark the constituents of all tied competitors
 end if
 while the highest-scoring hypothesis has a marked constituent do
  skip
 end while
 link the highest-scoring hypothesis
 discard all incompatible hypotheses
 unmark all constituents
end while
```

a ⇔ b	score_1
a ⇔ c	score_1
d ⇔ c	score_2
e ⇔ f	score_3
g ⇔ b	score_4
h ⇔ i	score_4

skip2

```
while unprocessed hypotheses with no tied competitors remain do
 if the highest-scoring hypothesis has tied competitors do
  mark the constituents of all tied competitors
 end if
 while the highest-scoring hypothesis has a marked constituent do
  skip
 end while
 link the highest-scoring hypothesis
 discard all incompatible hypotheses
 unmark all constituents
end while
```

a ⇔ b	score_1
a ⇔ c	score_1
d ⇔ c	score_2
e ⇔ f	score_3
g ⇔ b	score_4
h ⇔ i	score_4

skip2

```
while unprocessed hypotheses with no tied competitors remain do
 if the highest-scoring hypothesis has tied competitors do
  mark the constituents of all tied competitors
 end if
 while the highest-scoring hypothesis has a marked constituent do
  skip
 end while
 link the highest-scoring hypothesis
 discard all incompatible hypotheses
 unmark all constituents
end while
```

a ⇔ b	score_1
a ⇔ c	score_1
d ⇔ c	score_2
e ⇔ f	score_3
g ⇔ b	score_4
h ⇔ i	score_4

‘a ⇔ b’ is incompatible with ‘e ⇔ f’

skip2

```
while unprocessed hypotheses with no tied competitors remain do
 if the highest-scoring hypothesis has tied competitors do
  mark the constituents of all tied competitors
 end if
 while the highest-scoring hypothesis has a marked constituent do
  skip
 end while
 link the highest-scoring hypothesis
 discard all incompatible hypotheses
 unmark all constituents
end while
```

a ⇔ c	score_1
d ⇔ c	score_2
g ⇔ b	score_4
h ⇔ i	score_4

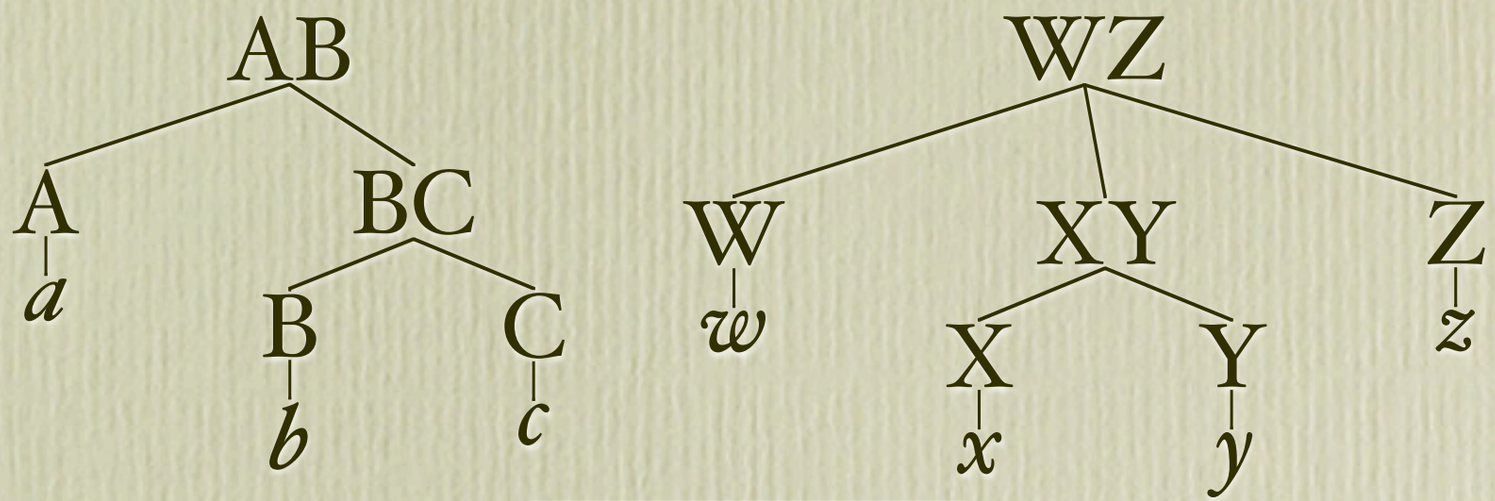
Spani

15/30

Span1

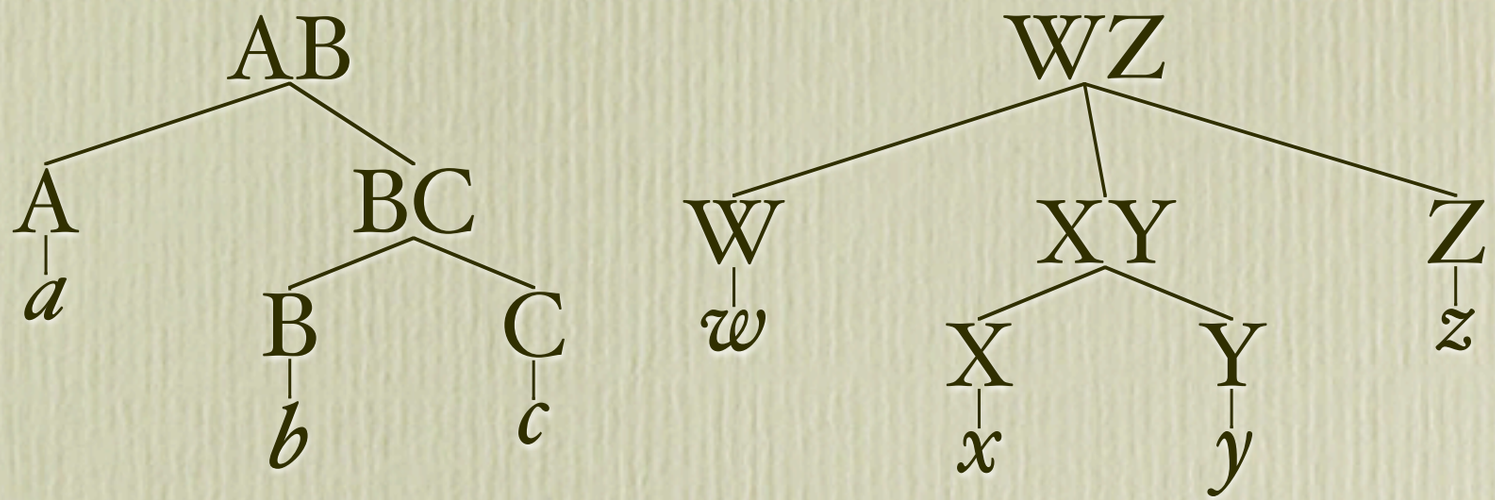
- High-scoring improper lexical links prevent the production of good non-lexical links
- Split the set of nonzero links into two sets:
  - lexical links
  - non-lexical links

Span



A	⇔	W
B	⇔	X
B	⇔	Z
C	⇔	Y
BC	⇔	XY
BC	⇔	Z
BC	⇔	WZ
AB	⇔	WZ

Span1



lexical

A	⇔	W
B	⇔	X
B	⇔	Z
C	⇔	Y
BC	⇔	Z

non-lexical

BC	⇔	XY
BC	⇔	WZ
AB	⇔	WZ

Span

- High-scoring improper lexical links prevent the production of good non-lexical links
- Split the set of nonzero links into two sets:
  - lexical links
  - non-lexical links
- Perform selection
  - first amongst the non-lexical links
  - then amongst the lexical links

Full-Search Selection

Full-Search Selection

- Backtracking Recursive Algorithm
  - enumerate all possible combinations of non-crossing links
    - store all maximal combinations of non-crossing links
- The best combination of links is the highest-scoring maximal combination of links
- Ambiguity again…

String-to-String Alignment

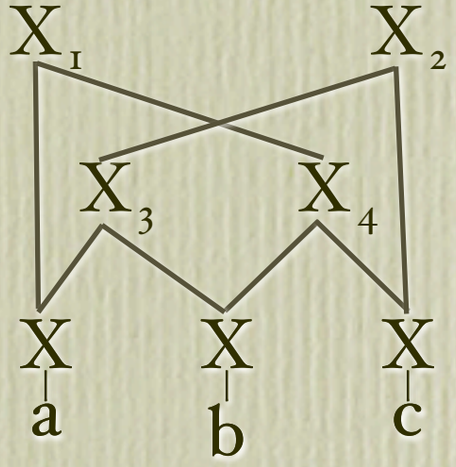
String-to-String Alignment

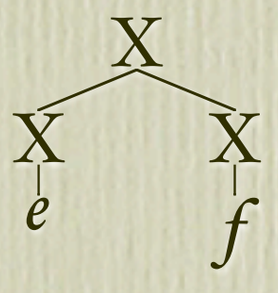
- The sub-tree aligner operates on parsed data
- For many languages no parsers are available
  - Retraining existing parsers for new languages may require significant resources
- The string-to-string aligner operates on plain sentences

String-to-String Alignment algorithm

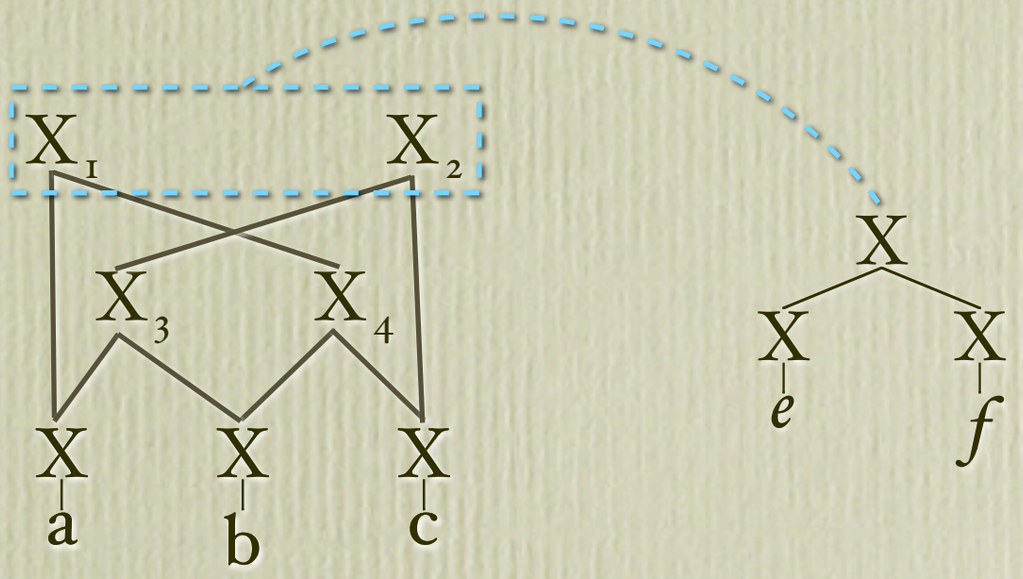
- Generate all possible binary trees for each sentence in the sentence pair
- Calculate scores for all possible links
- Select the best set of links as before
    - Two links are incompatible if they contain nodes that are part of incompatible trees

String-to-String Alignment algorithm

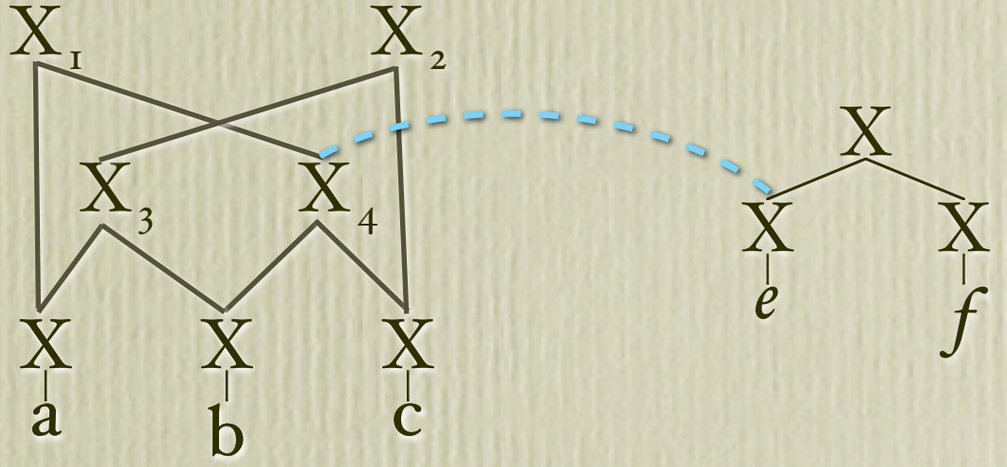




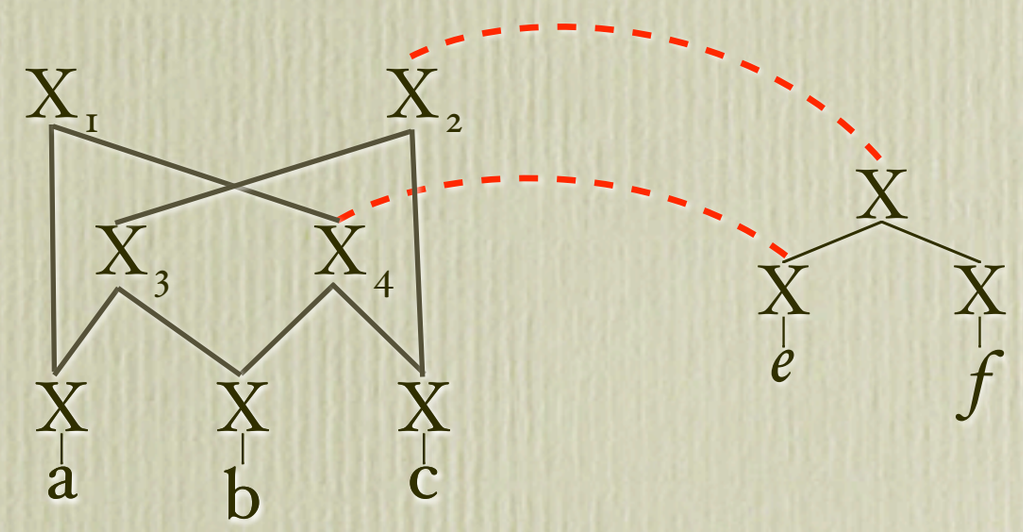
String-to-String Alignment algorithm



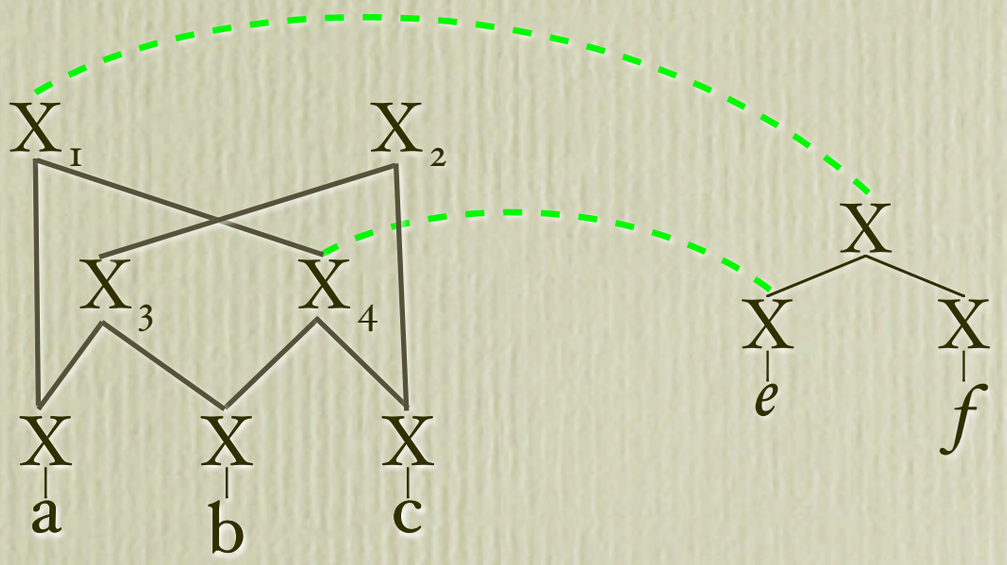
String-to-String Alignment algorithm



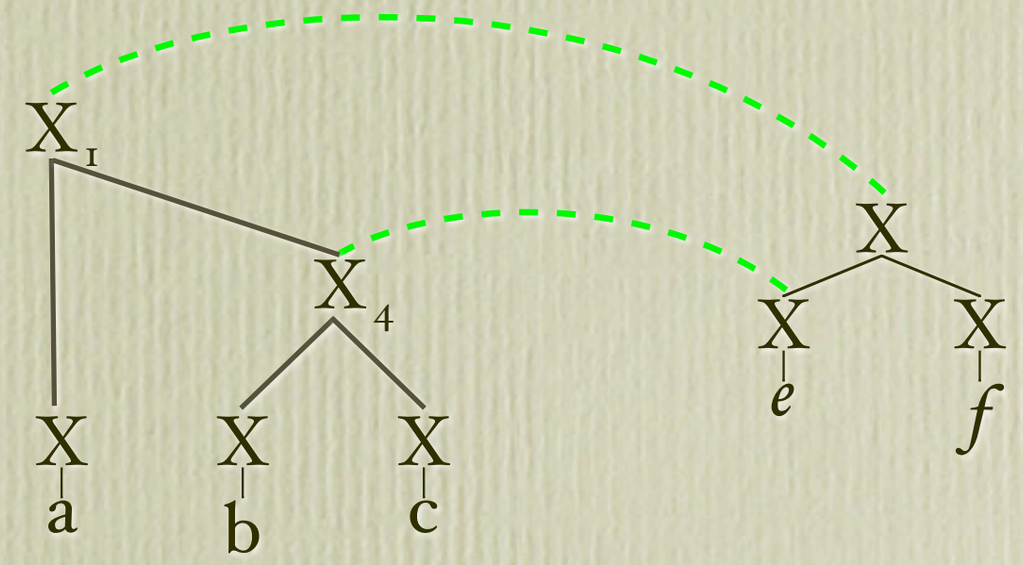
String-to-String Alignment algorithm



String-to-String Alignment algorithm



String-to-String Alignment algorithm



String-to-String Alignment algorithm

- Generate all possible binary trees for each sentence in the sentence pair
- Calculate scores for all possible links
- Select the best set of links as before
  - Two links are incompatible if they contain nodes that are part of incompatible trees
- Only output linked nodes
  - and nodes needed for structural integrity

String-to-String Alignment

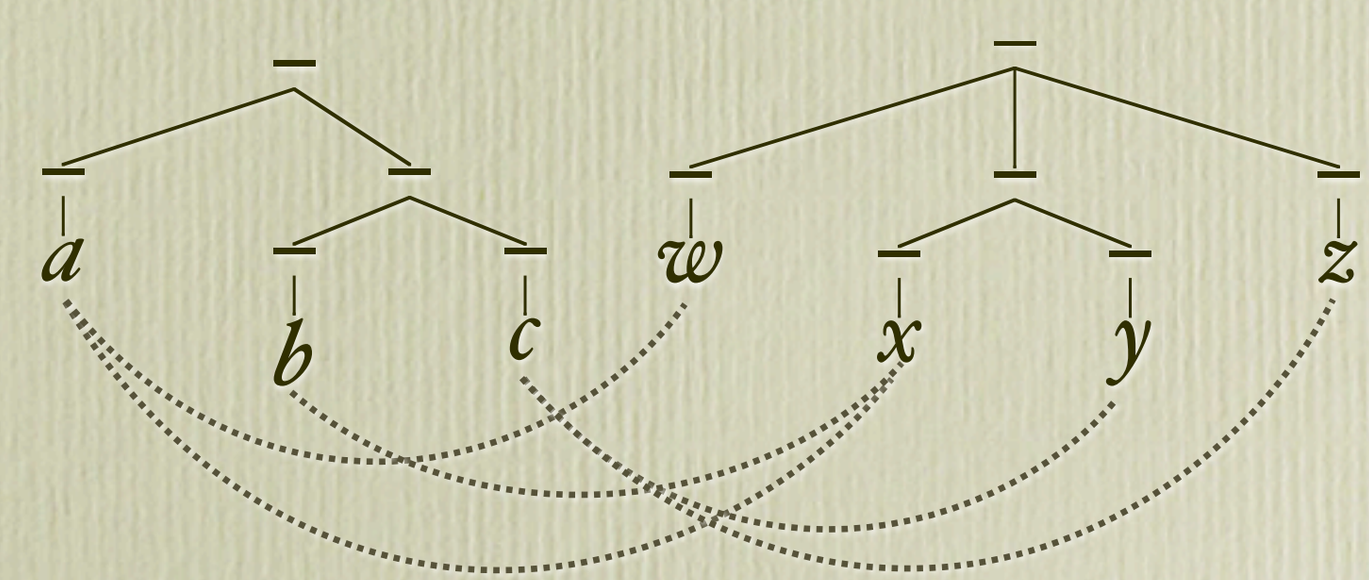
- Tree-to-string and string-to-tree modules
  - used when a parser exists for one of the languages in question
  - based on the string-to-string module
  - preserve original parse structures

Re-Scoring Algorithm

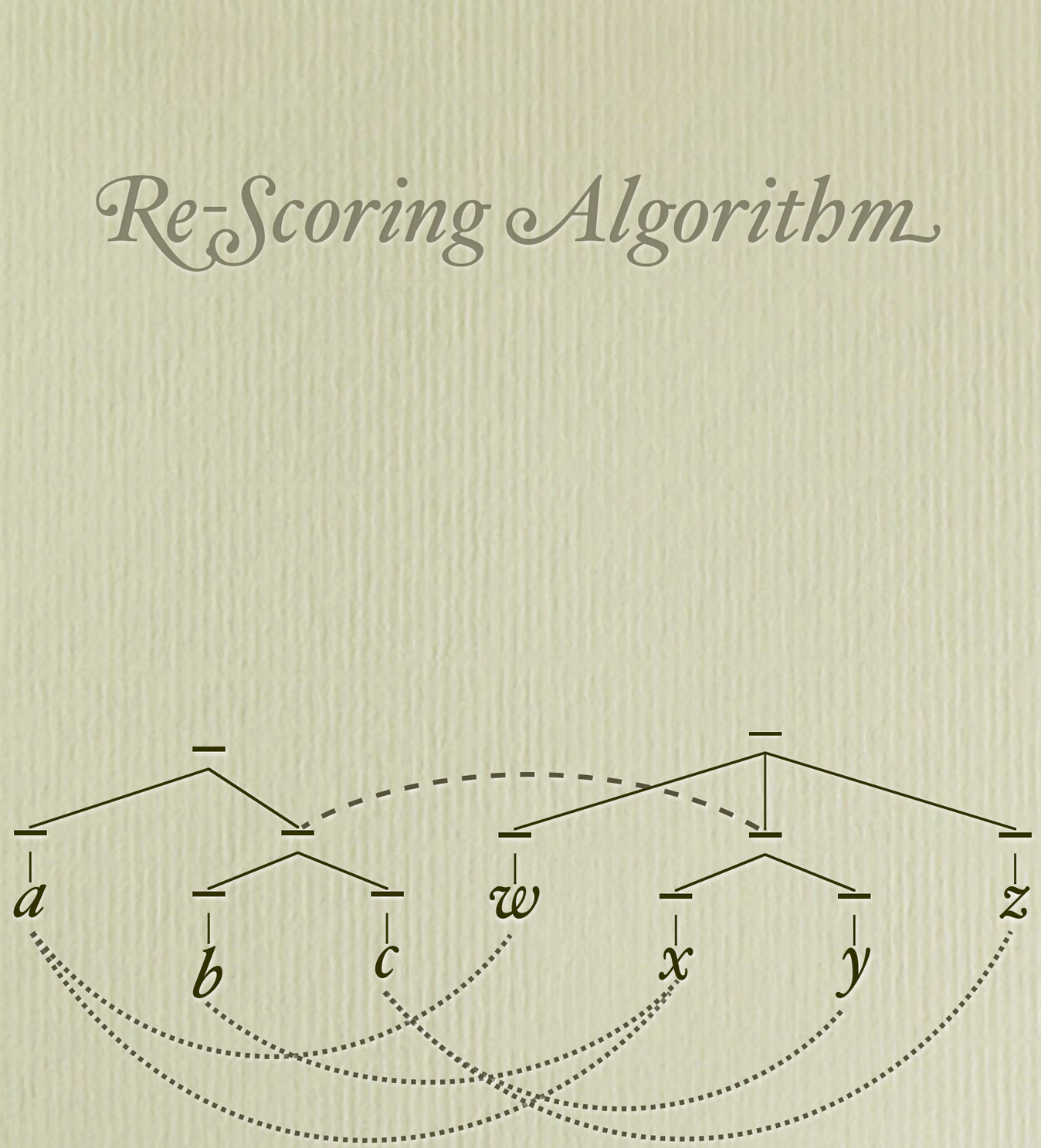
Re-Scoring Algorithm

- Use already fixed links to establish the context for word alignments

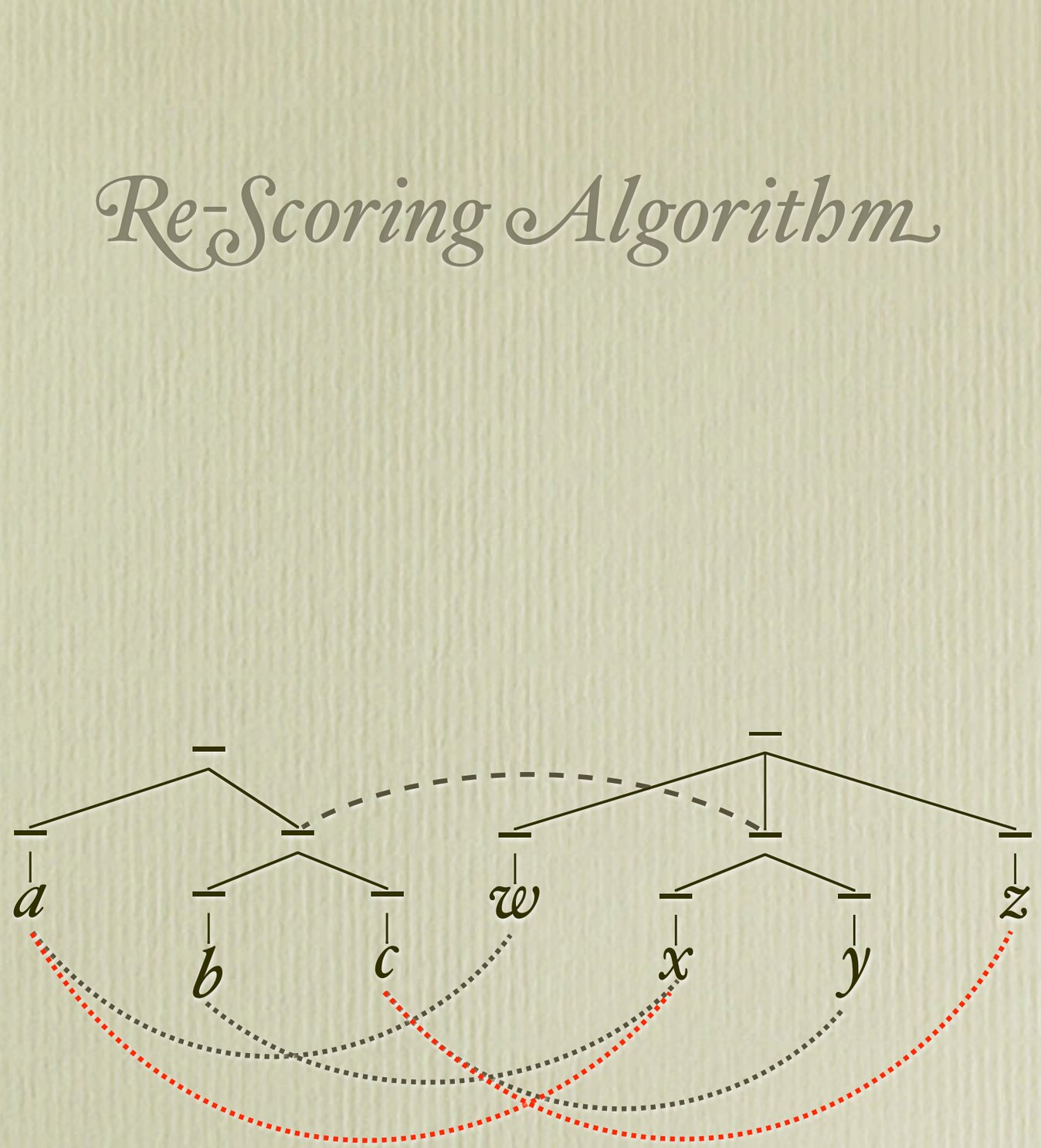
Re-Scoring Algorithm



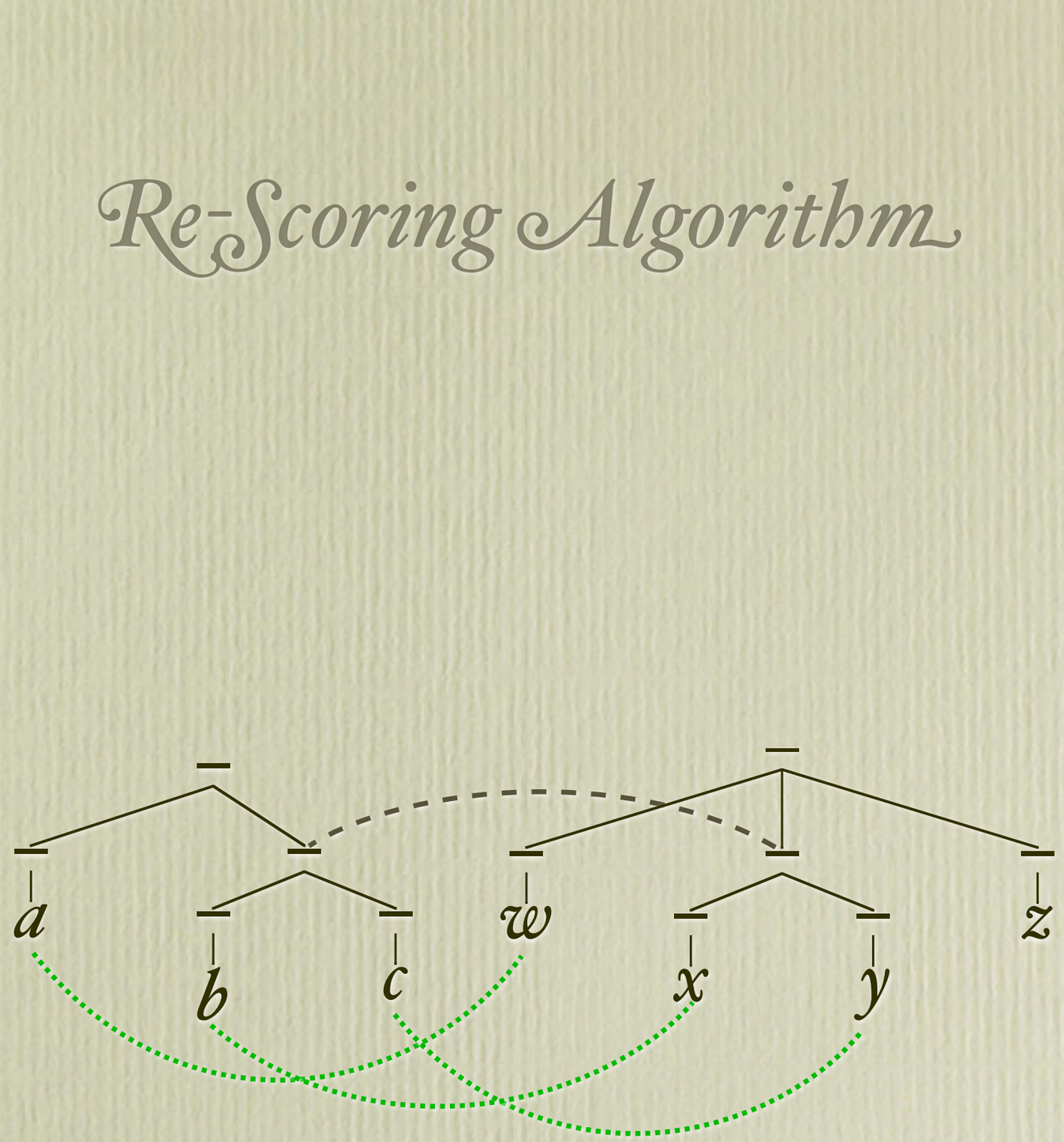
Re-Scoring Algorithm



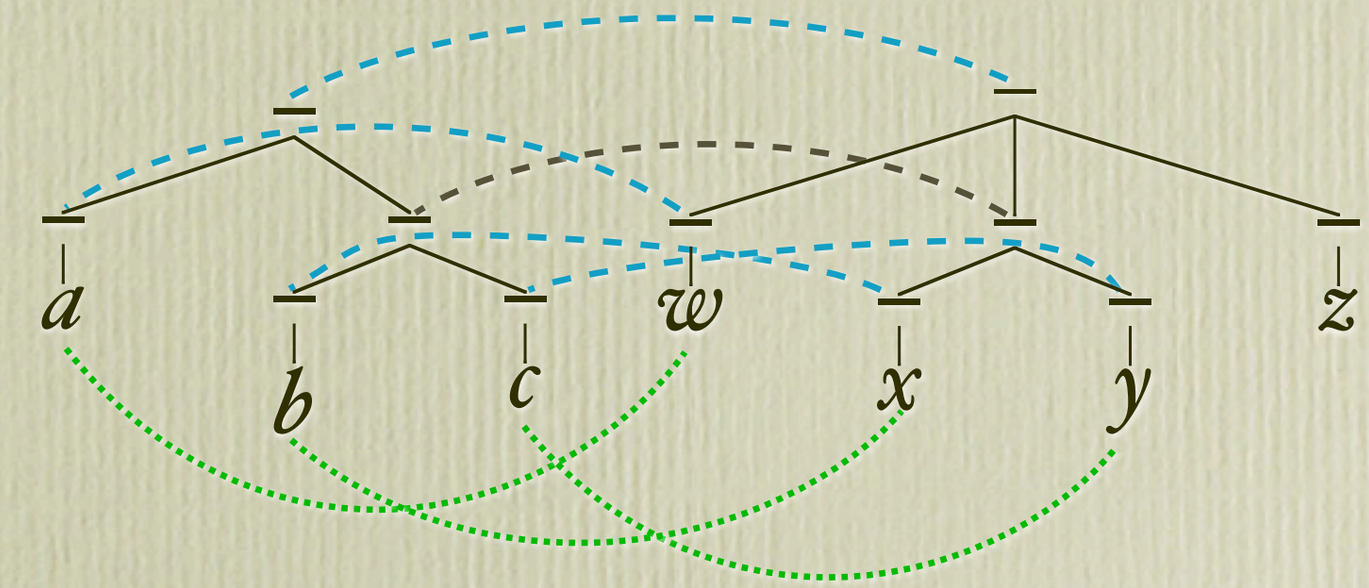
Re-Scoring Algorithm



Re-Scoring Algorithm



Re-Scoring Algorithm



Algorithm Complexity

Algorithm Complexity

- Tree-to-Tree Alignment
  - space: $O(n)$
  - time: $O(m^2)$
- String-to-String Alignment
  - space: $O(n^2)$
  - time: $O(m^4)$
- Full-Search Algorithm
  - related to the TSP, albeit highly restricted
- Score calculation may be executed in parallel

Contents of the Distribution

Contents of the Distribution

- Available at
  `http://ventsislavzhechev.eu/Home/Software/Software.html`
  - an RSS feed is available for update notifications
    `http://ventsislavzhechev.eu/Home/Software/rss.xml`
- GPL licence
- `README` file
- C++ code of the sub-tree aligner
- Configuration and compilation scripts

Required Software

Required Software

- GCC 4.0+
  - GCC 4.2+ required for the compilation of parallel code
- boost 1.34+ for string-to-string, tree-to-string and string-to-tree modules
- Should compile on any UNIX system

Avenues for Improvement

Avenues for Improvement

- Add new scoring algorithms
  (eg. maximum-entropy-based)
- Research ways to reduce the number of scores
  that need to be calculated
- Store word-alignment data in memory
  in a more optimal way
- Research ways to integrate POS-tag data into the
  structures generated by the string-based modules
- Make the main features selectable at run time,
  rather than at compile time

Conclusions

Conclusions

- Developed a novel platform for the fast and robust generation of parallel treebanks
- The aligner can handle very large amounts of data
- There are many underresourced languages
  - String-to-string, string-to-tree and tree-to-string algorithms have been developed
- Please send your bug reports to **`bugs@ventsislavzhechev.eu`**

Thank you!