Incremental Dependency Parsing

Michael Fell

9 June 2011

UNIVERSITÄT
DES
SAARLANDES

Computational
Linguistics
& Phonetics

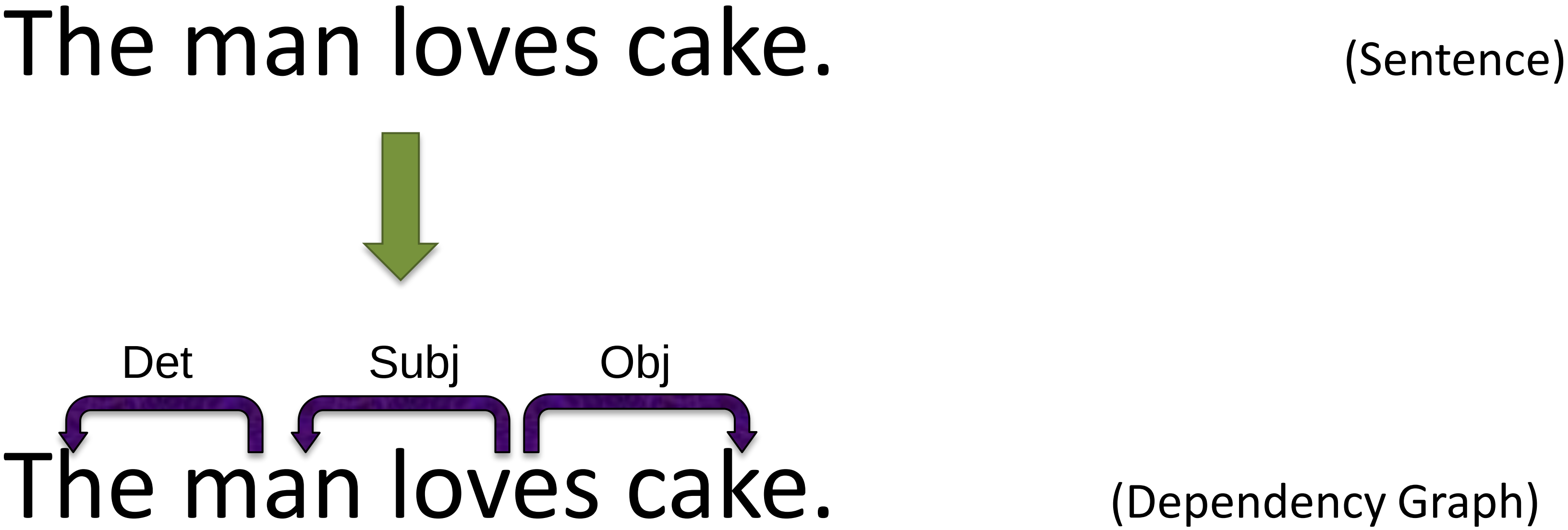
Overview

- Incremental Dependency Parsing
  - two algorithms
  - evaluation

- General criticism on present approaches
  - possible improvements

- Summary

Dependency Parsing



The man loves cake. (Sentence)

The man loves cake. (Dependency Graph)

Dependency Graph

Labeled, directed graph (W, A)

- W: words in the sentence
- A: dependency relation between words

Dependency Graph

Labeled, directed graph (W, A)

- W: words in the sentence
- A: dependency relation between words

Well-formedness criteria:

- connected
- acyclic
- unique label
- single head
- projective

Incremental Dependency Parsing

Dependency parsing

- is robust and performs well
- omits phrasal nodes

→What about doing it incrementally?

Incremental Dependency Parsing

Dependency parsing

- is robust and performs well
- omits phrasal nodes

→What about doing it incrementally?

→One possibility:

Left-to-right bottom-up dependency parsing

Bottom-up Dependency Parsing

stack

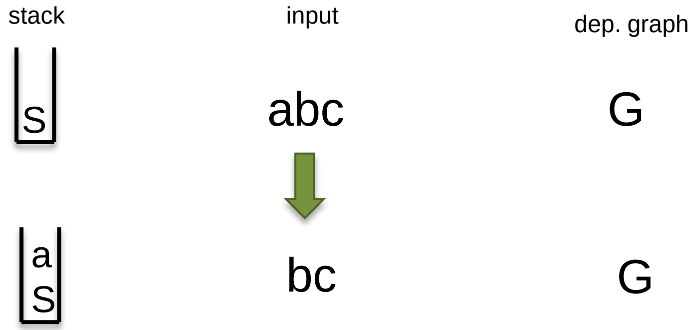
input

dep. graph

Bottom-up Dependency Parsing

Shift:

(S)

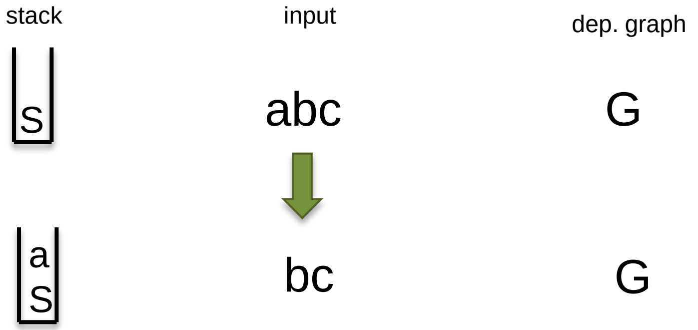


stack	input	dep. graph
S	abc	G
a S	bc	G

Bottom-up Dependency Parsing

Shift:

(S)



stack

input

dep. graph

S

abc

G

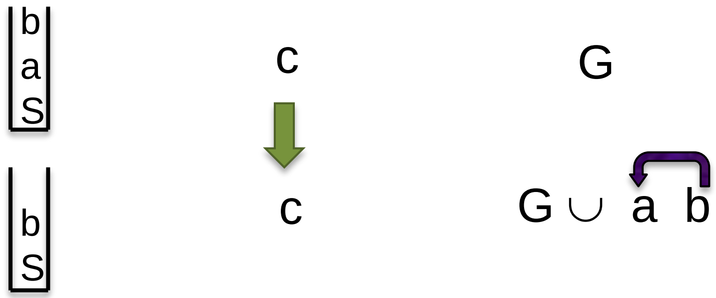
a

S

bc

G

Left-Reduce (LR):



b

a

S

c

G

b

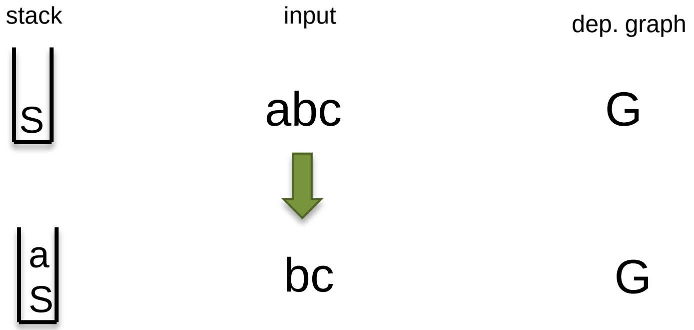
S

c

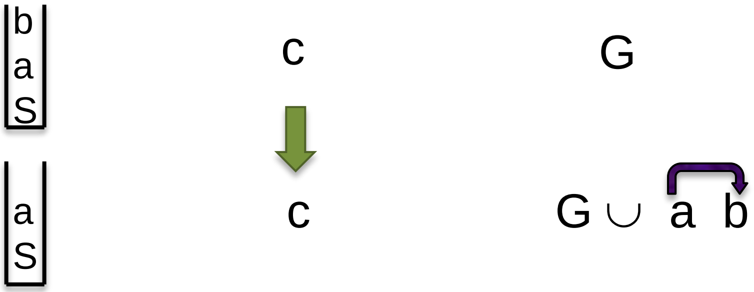
G ∪ a b

Bottom-up Dependency Parsing

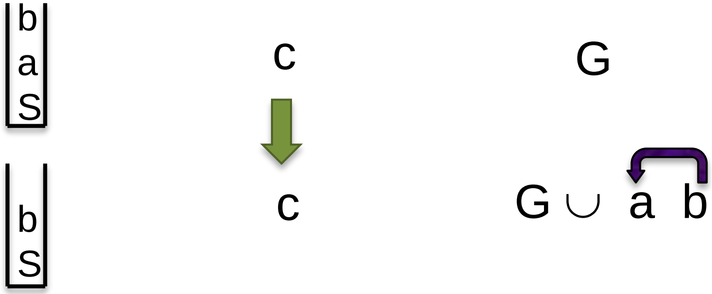
Shift:
(S)



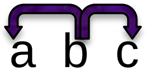
Right-Reduce (RR):

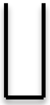


Left-Reduce (LR):



Bottom-up Dependency Parsing

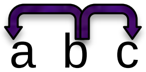
Example derivation of 



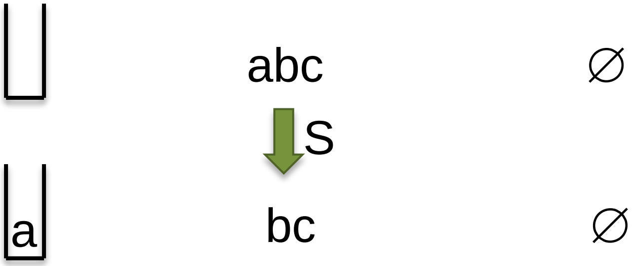
abc ∅

Bottom-up Dependency Parsing

Example derivation of



a b c

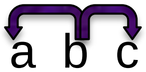


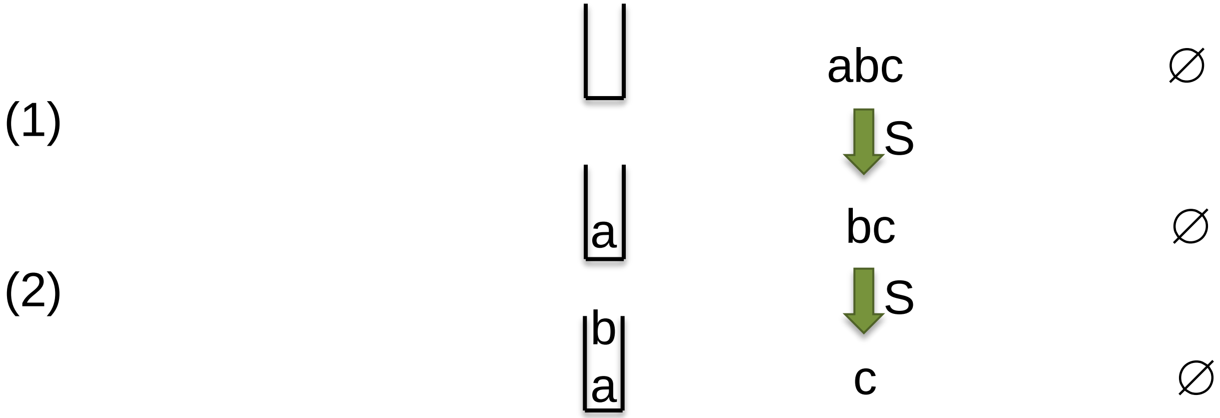
abc ∅

(1) S

a bc ∅

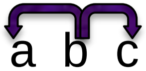
Bottom-up Dependency Parsing

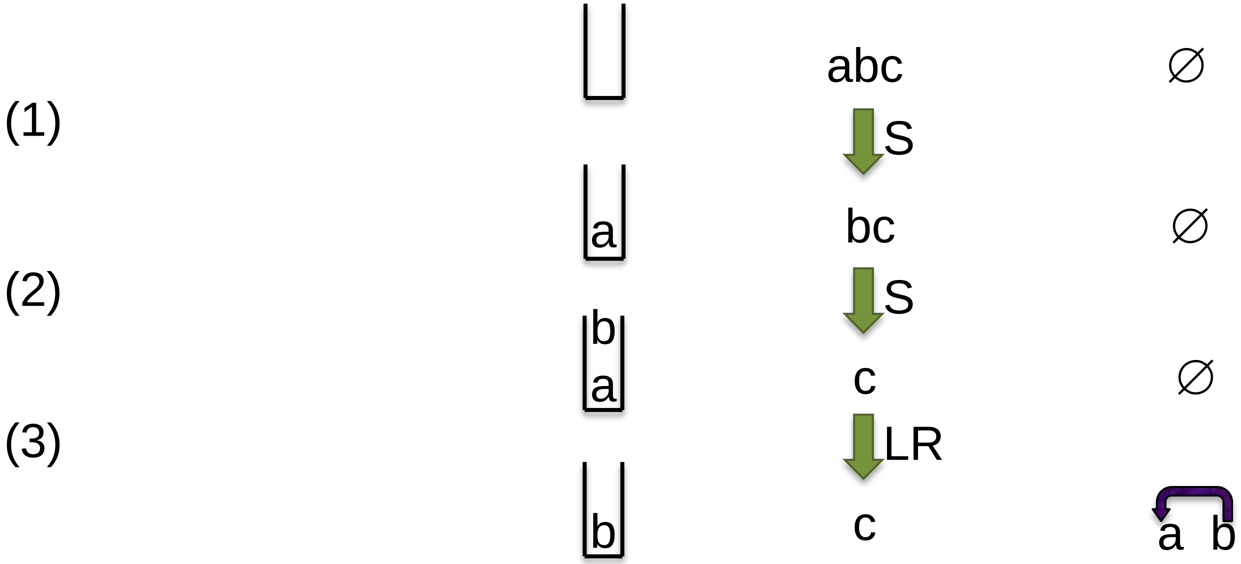
Example derivation of 



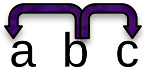
		abc	∅
(1)		S	
	a	bc	∅
(2)		S	
	b a	c	∅

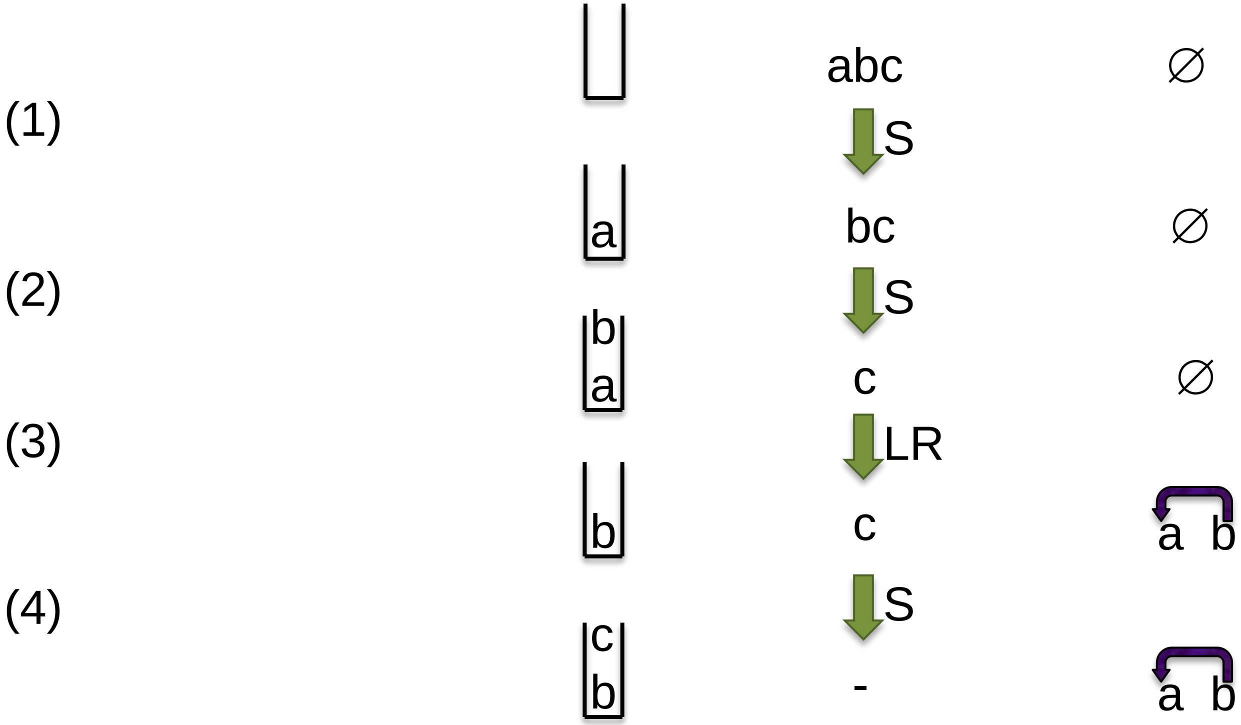
Bottom-up Dependency Parsing

Example derivation of 



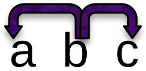
Bottom-up Dependency Parsing

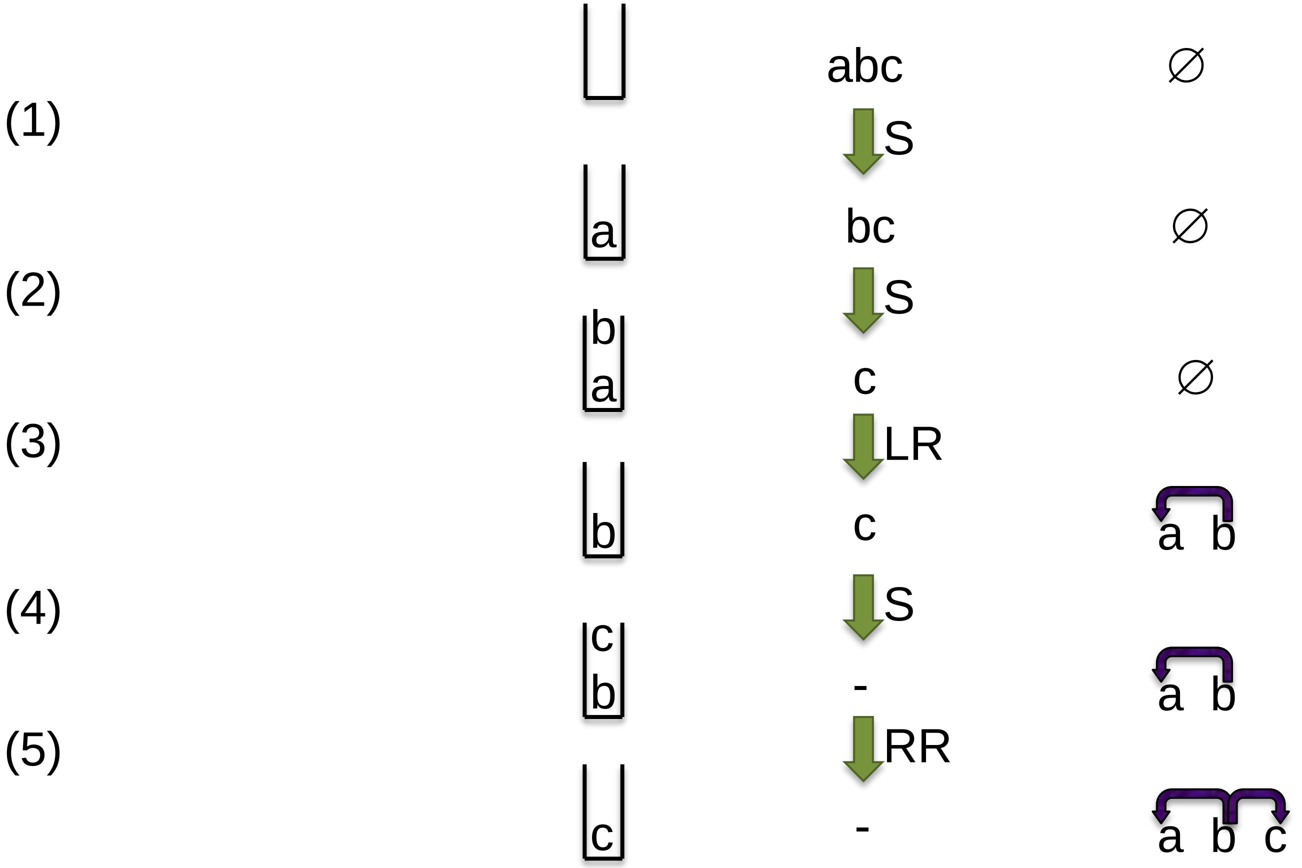
Example derivation of 



Bottom-up dependency parsing

Example derivation of





abc ∅

(1) S

a bc ∅

(2) S

b a c ∅

(3) LR

b c a b

(4) S

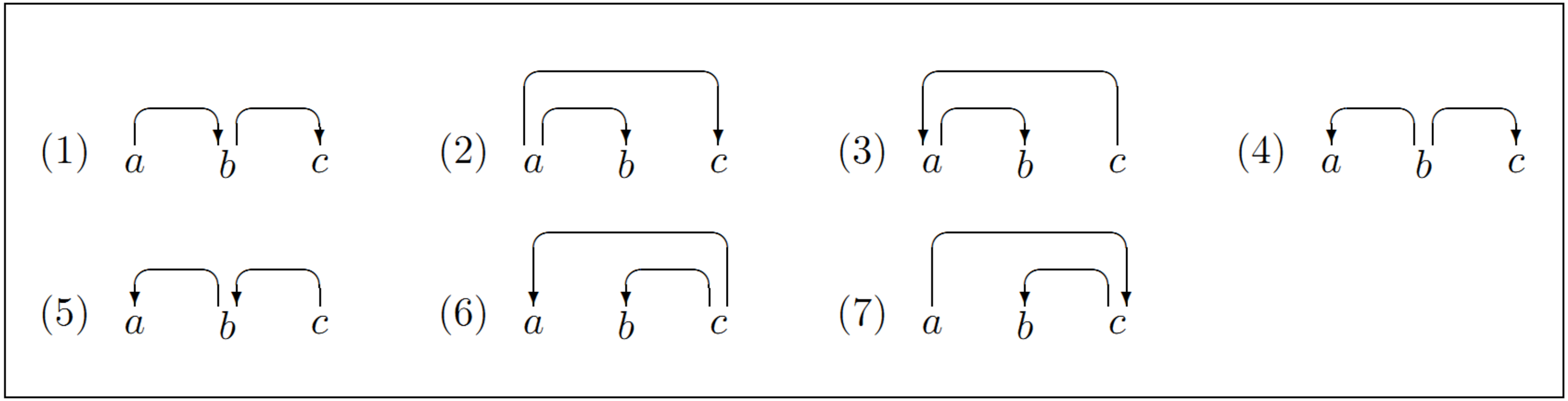
c b - a b

(5) RR

c - a b c

Bottom-up dependency parsing

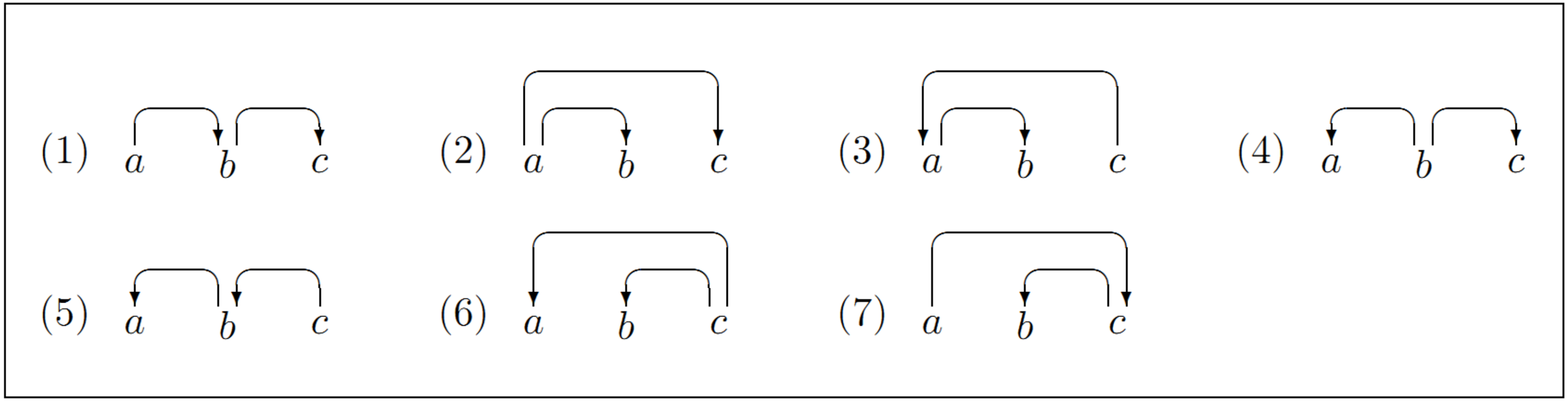
Dependency graphs with 3 nodes:



We have derived (4). (2), (3) and (5) can also be derived.

Bottom-up dependency parsing

Dependency graphs with 3 nodes:

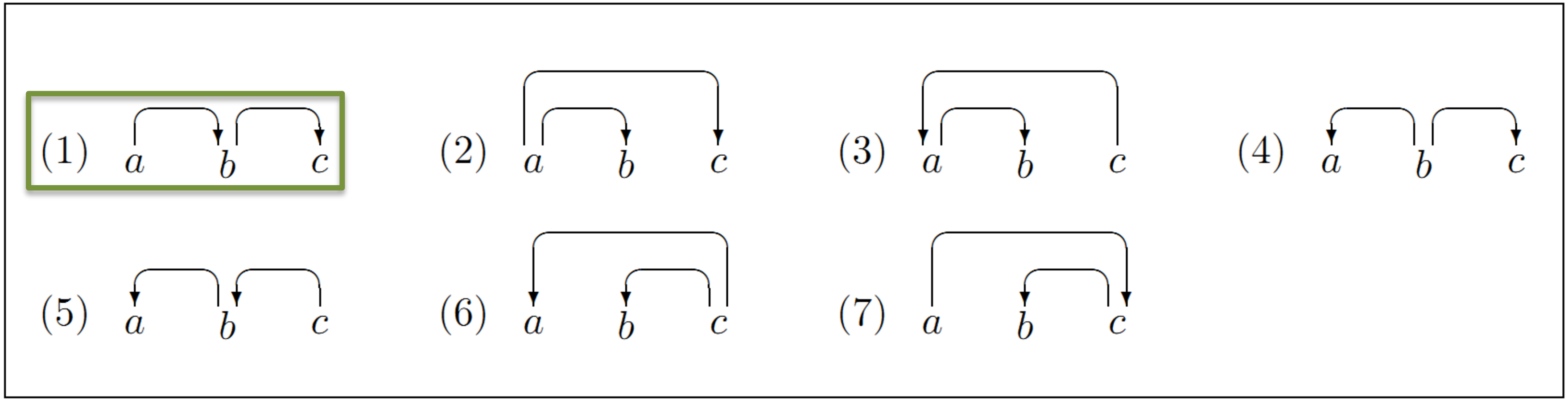


We have derived (4). (2), (3) and (5) can also be derived.

(1) and (6), (7) can’t be derived

Bottom-up Dependency Parsing

Dependency graphs with 3 nodes:



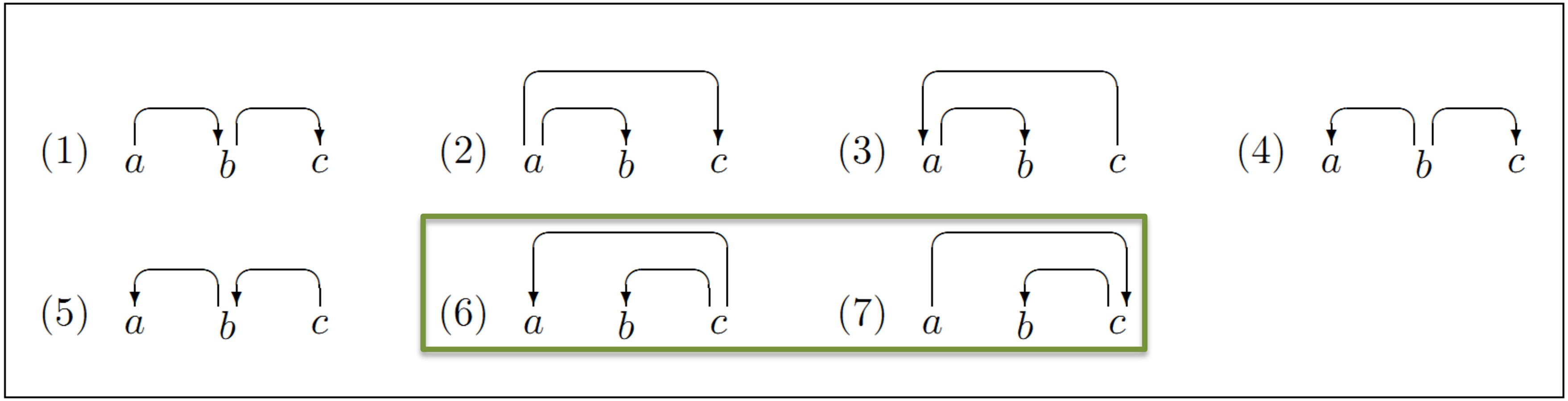
We have derived (4). (2), (3) and (5) can also be derived.

(1) and (6), (7) can’t be derived

(1): b is combined via Right-Reduction → b has a head → b erased from stack

Bottom-up Dependency Parsing

Dependency graphs with 3 nodes:



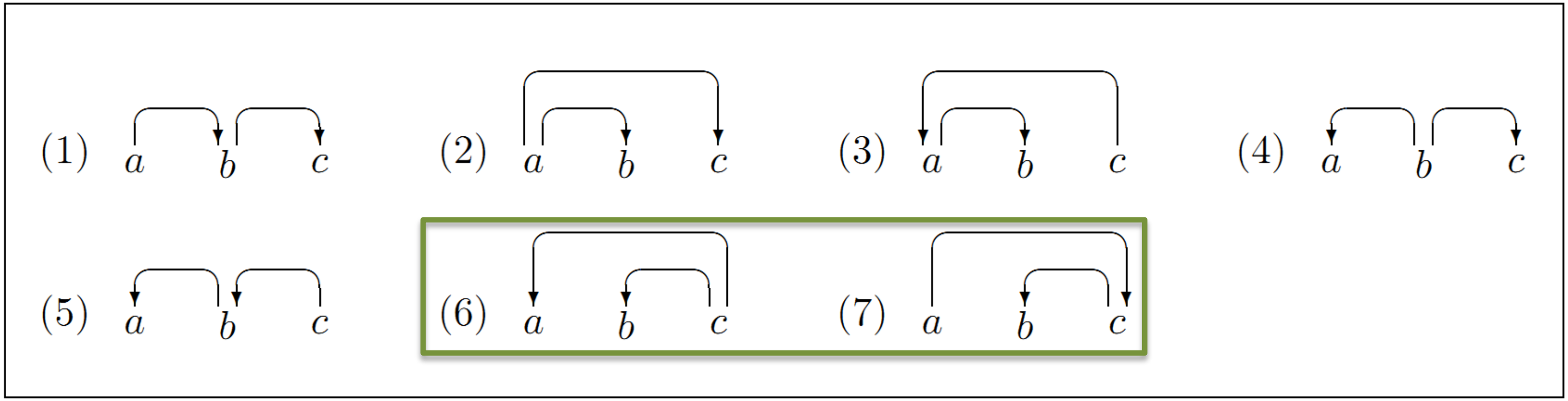
We have derived (4). (2), (3) and (5) can also be derived.

(1) and (6), (7) can’t be derived

(1): b is combined with a via Right-Reduction → b has a head → b erased from stack

(6), (7): no connecting arc between a and b → To connect them, we needed to put c onto the stack, too. (hence lose incrementality)

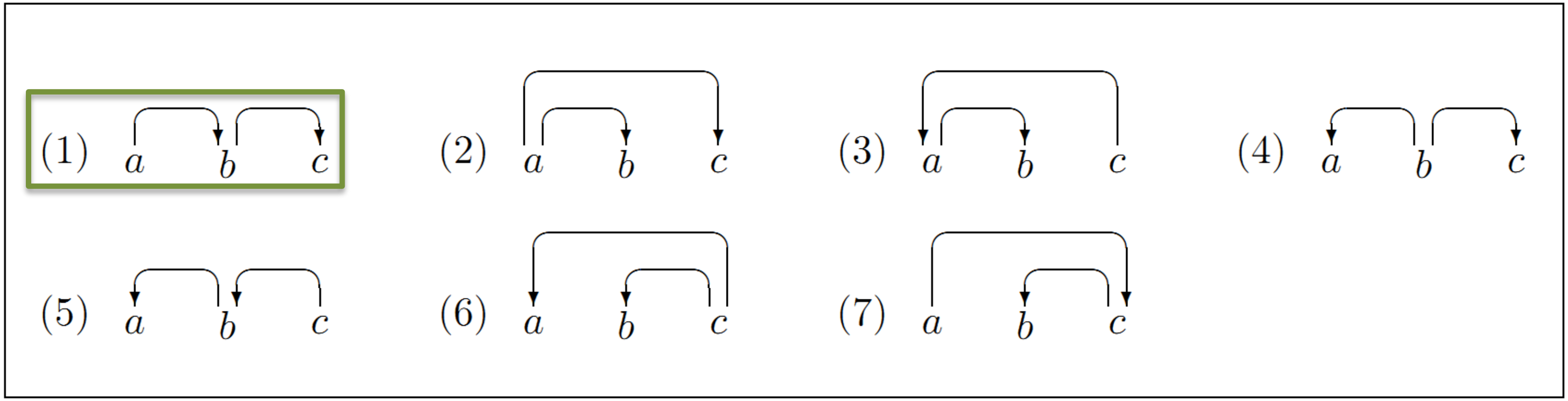
Bottom-up Dependency Parsing



Is there a way to parse (1) and (6), (7) incrementally?

→(6), (7): no!

Bottom-up Dependency Parsing



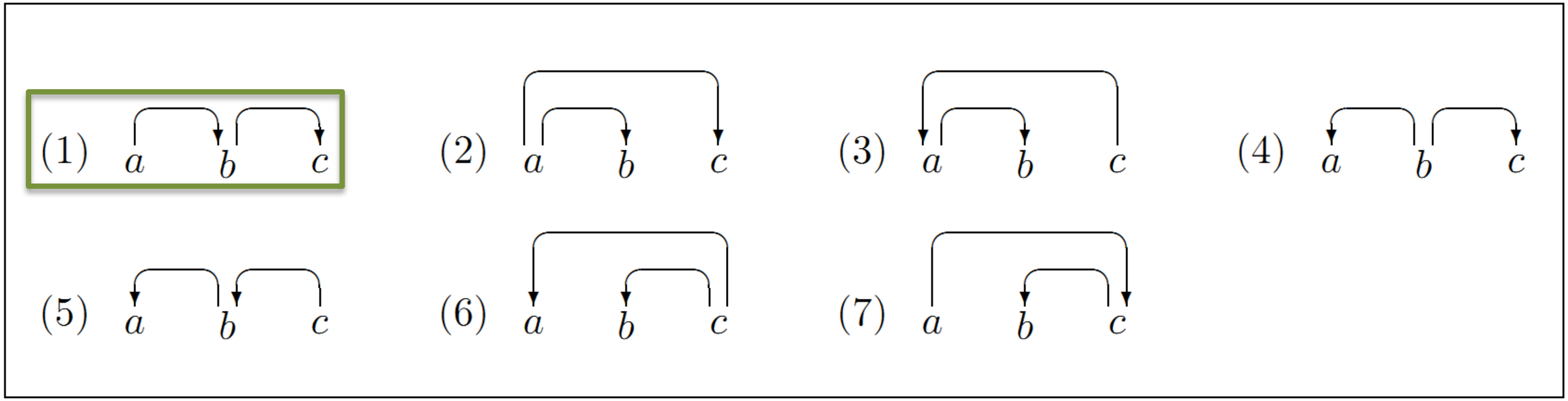
Is there a way to parse (1) and (6), (7) incrementally?

→(6), (7): no!

→(1): yes, read input from right to left

- →incrementality?

Bottom-up Dependency Parsing



Is there a way to parse (1) and (6), (7) incrementally?

→(6), (7): no!

→(1): yes, read input from right to left

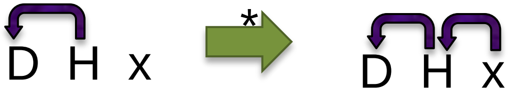
- →incrementality?

(1) can be processed incrementally

Incremental Dependency Parsing

Bottom-up and Top-down in Dependency Parsing

BU:

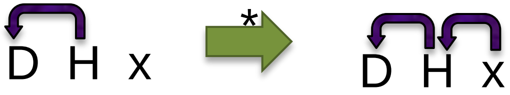


Dependent D is attached to its head H before H is attached to its head

Incremental Dependency Parsing

Bottom-up and Top-down in Dependency Parsing

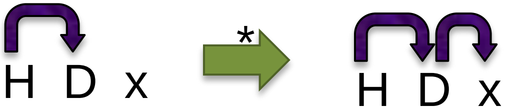
BU:



D H x * D H x

Dependent D is attached to its head H before H is attached to its head

TD:



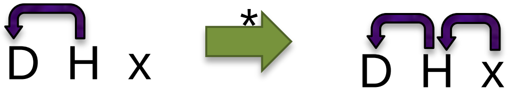
H D x * H D x

Head H is attached to a dependent D before D is attached to its dependent(s)

Incremental Dependency Parsing

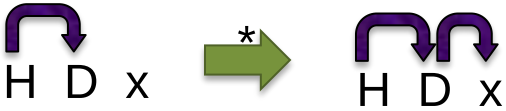
Bottom-up and Top-down in Dependency Parsing

BU:



Dependent D is attached to its head H before H is attached to its head

TD:



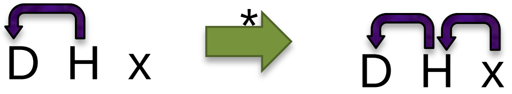
Head H is attached to a dependent D before D is attached to its dependent(s)

Insight:
→ We can process left-dependents incrementally via BU
→ process right-dependents incrementally via TD parsing

Incremental Dependency Parsing

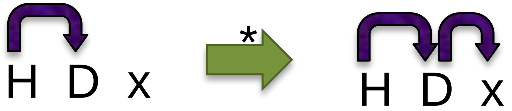
Bottom-up and Top-down in Dependency Parsing

BU:



Dependent D is attached to its head H before H is attached to its head

TD:



Head H is attached to a dependent D before D is attached to its dependent(s)

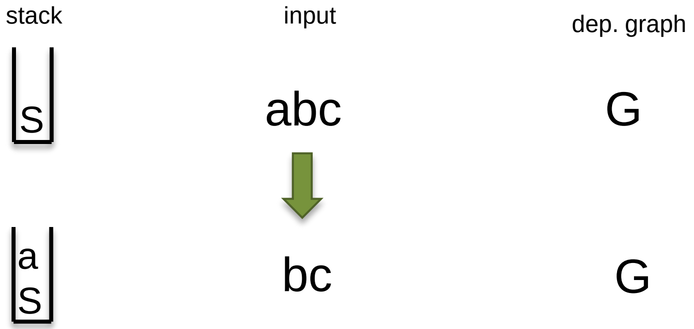
Insight:
→ We can process left-dependents incrementally via BU
→ process right-dependents incrementally via TD parsing

→ Arc-Eager Dependency Parsing

Arc-Eager Dependency Parsing

Shift:

(S)



stack

input

dep. graph

S

abc

G

a

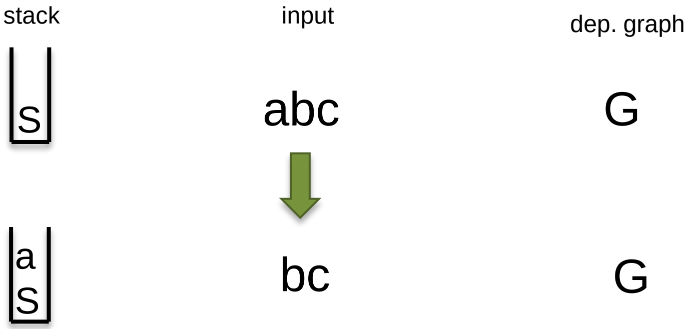
S

bc

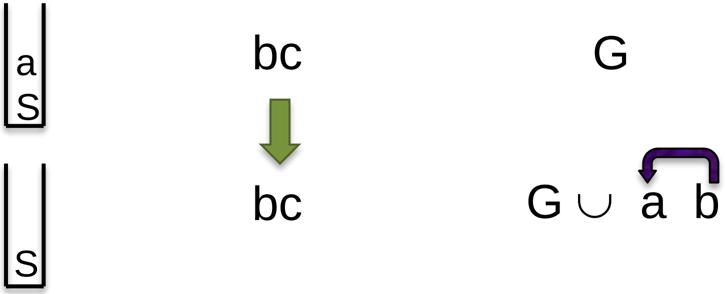
G

Arc-Eager Dependency Parsing

Shift:
(S)



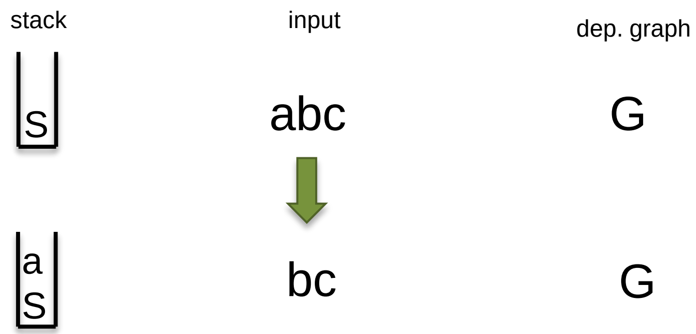
Left-Arc (LA):



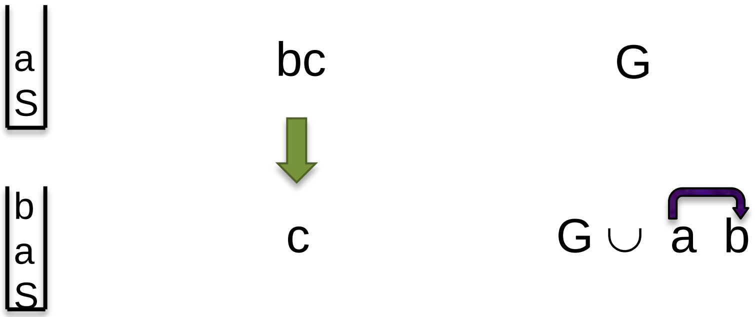
Arc-Eager Dependency Parsing

Shift:

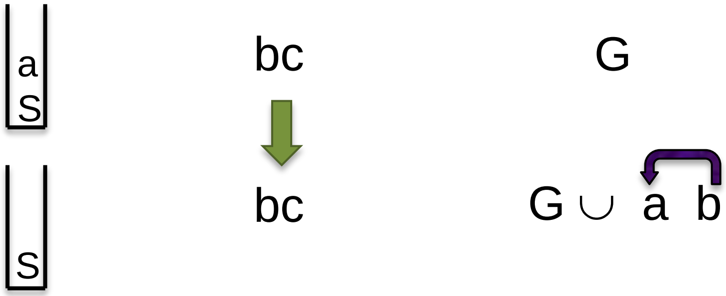
(S)



Right-Arc (RA):



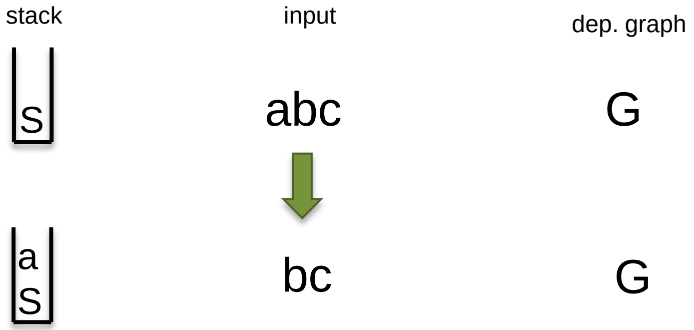
Left-Arc (LA):



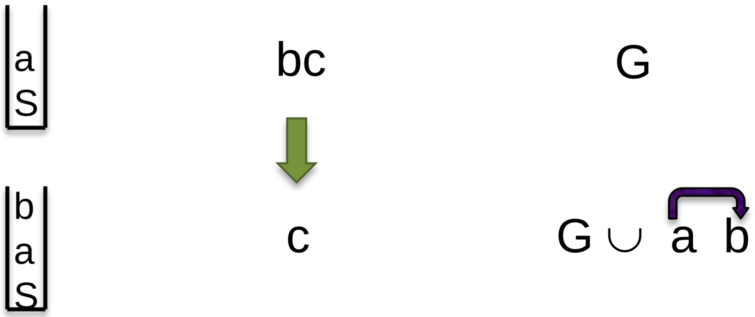
Arc-Eager Dependency Parsing

Shift:

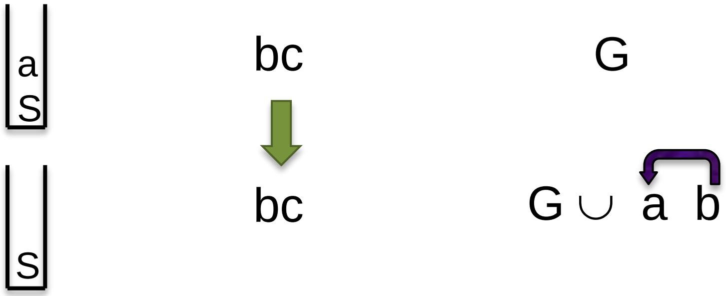
(S)



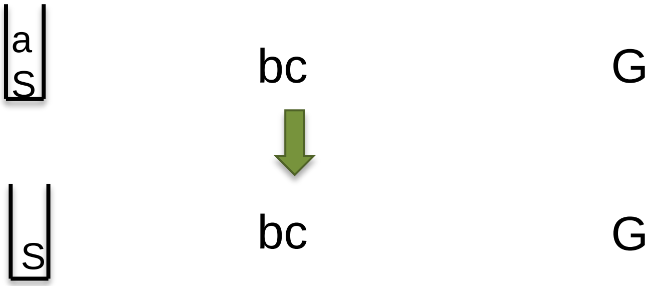
Right-Arc (RA):



Left-Arc (LA):



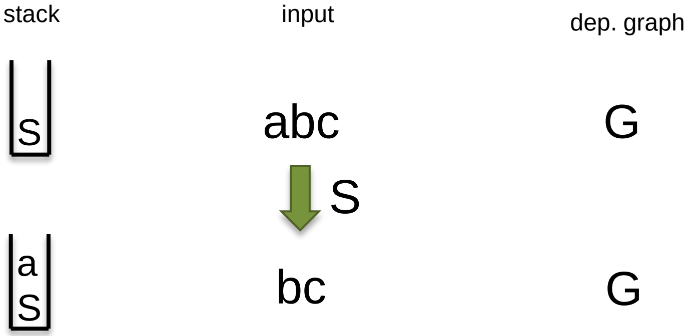
Reduce (R):



Bottom-up vs. Arc-Eager

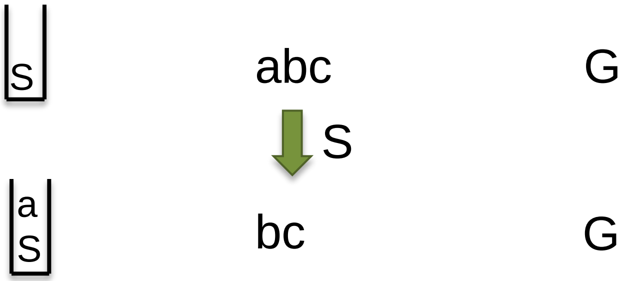
Shift:

(S_{BU})



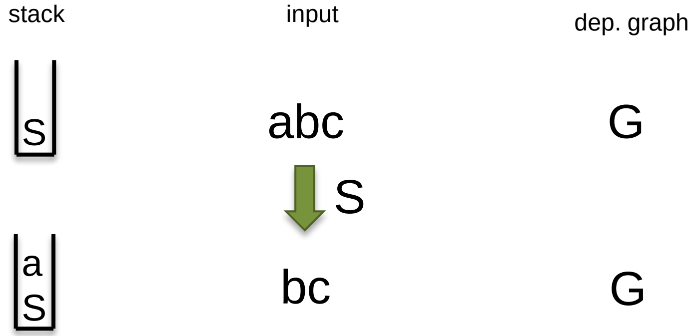
Shift:

(S_{AE})

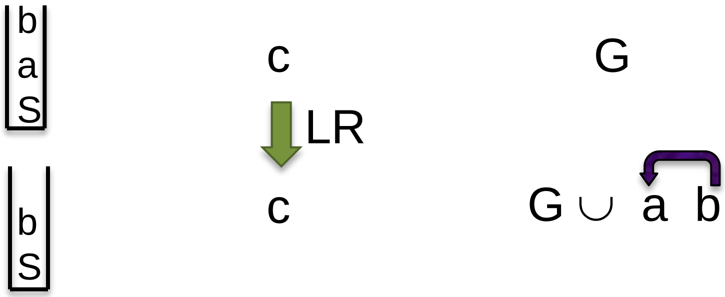


Bottom-up vs. Arc-Eager

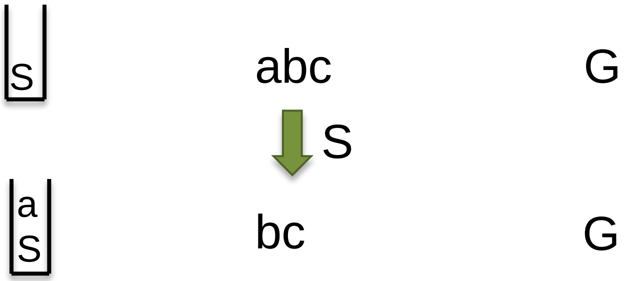
Shift: (S_{BU})



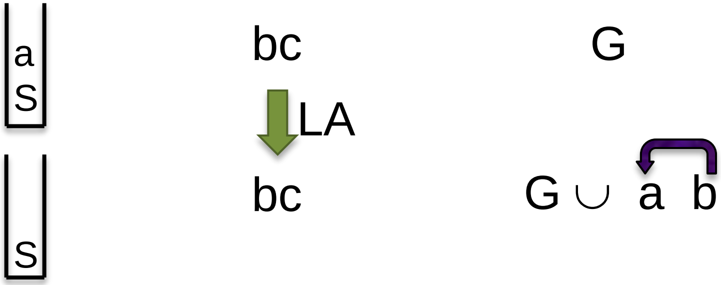
Left-Reduce (LR):



Shift: (S_{AE})



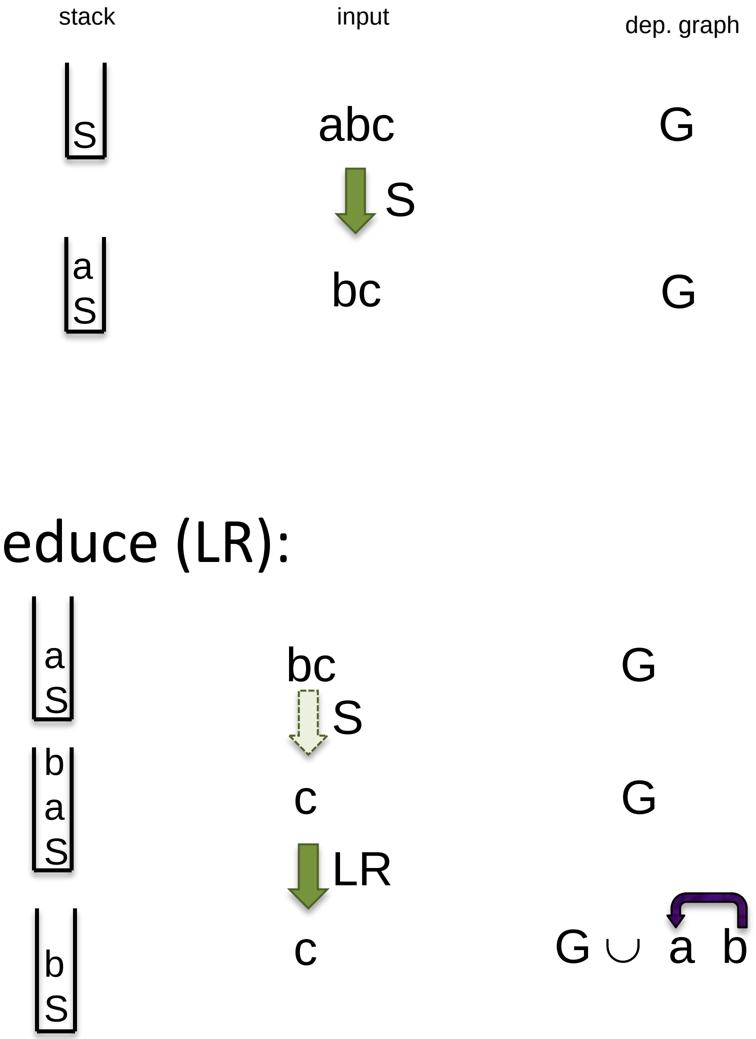
Left-Arc (LA):



Bottom-up vs. Arc-Eager

Shift:

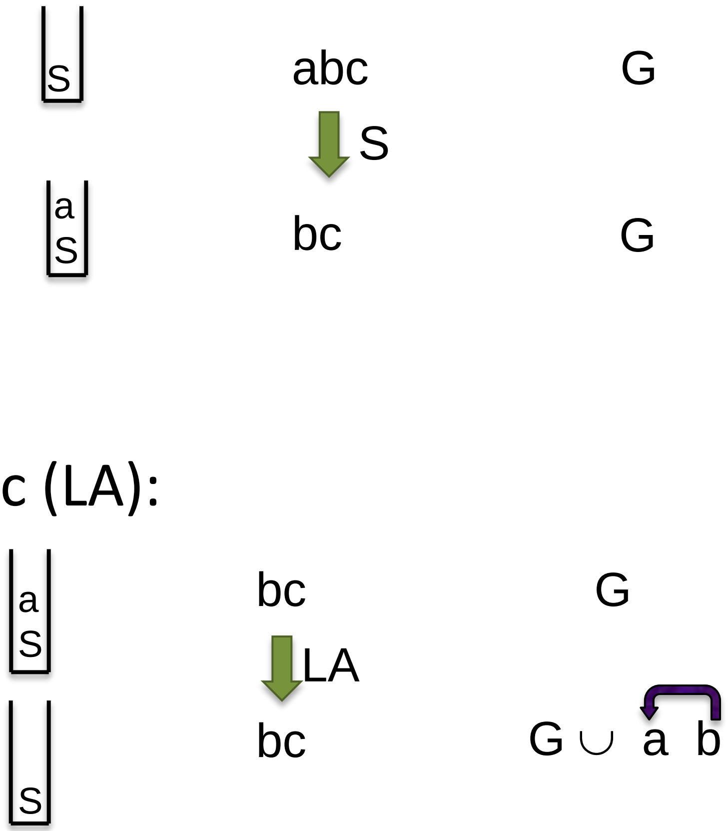
(S_{BU})



Left-Reduce (LR):

Shift:

(S_{AE})

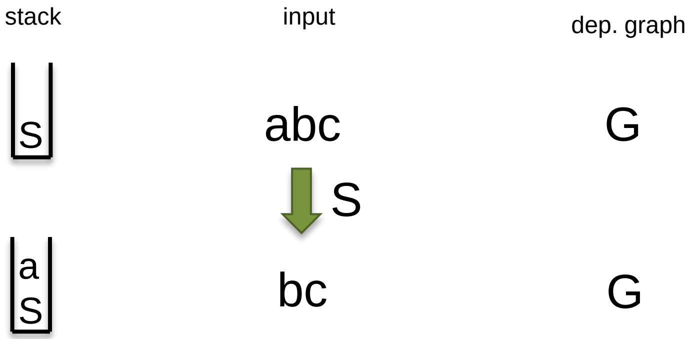


Left-Arc (LA):

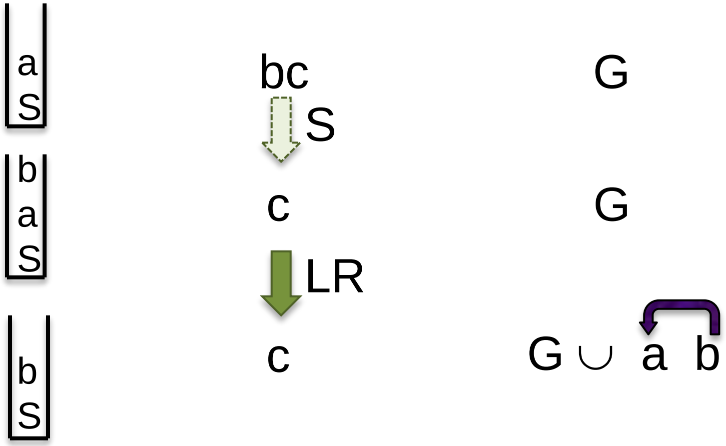
Bottom-up vs. Arc-Eager

Shift:

(S_{BU})

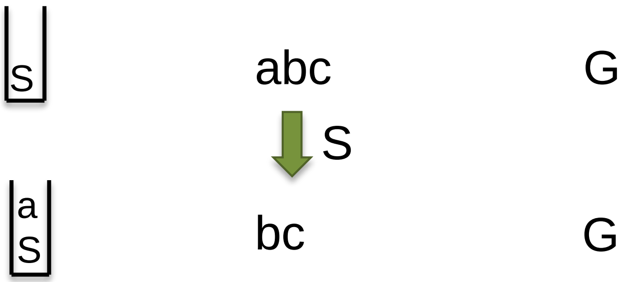


Left-Reduce (LR):

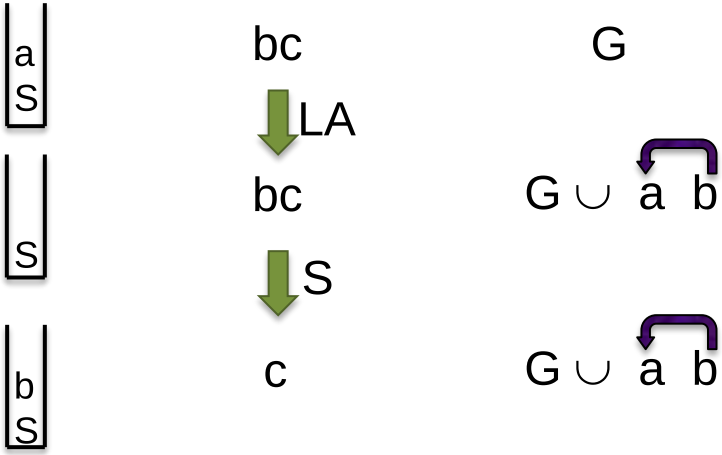


Shift:

(S_{AE})

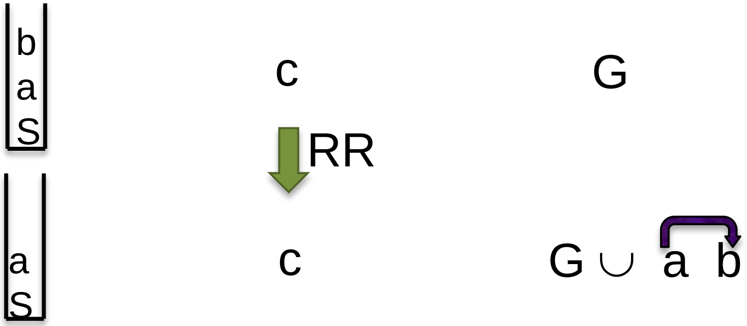


Left-Arc (LA):



Bottom-up vs. Arc-Eager

Right-Reduce (RR):

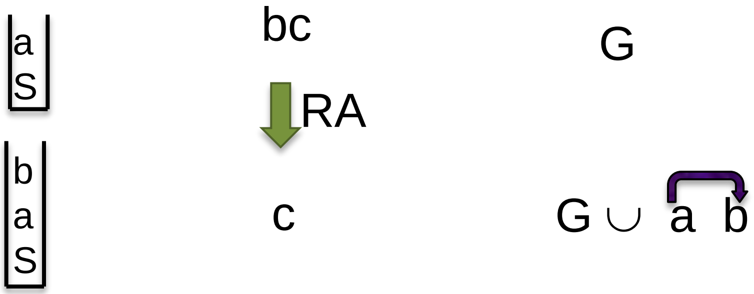


b a S | c | G

RR

a S | c | G ∪ a b

Right-Arc (RA):



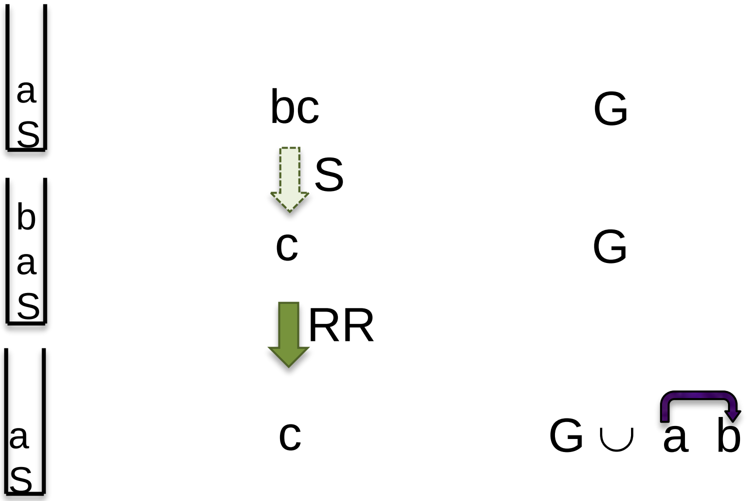
a S | bc | G

RA

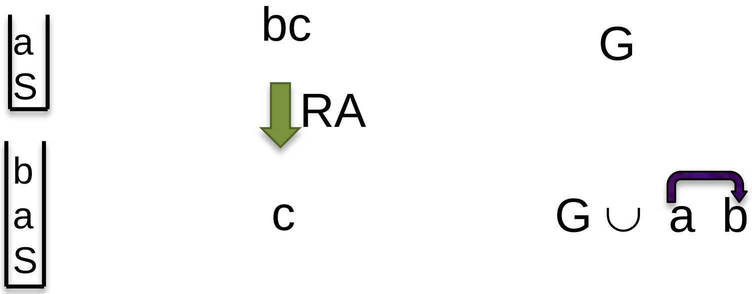
b a S | c | G ∪ a b

Bottom-up vs. Arc-Eager

Right-Reduce (RR):

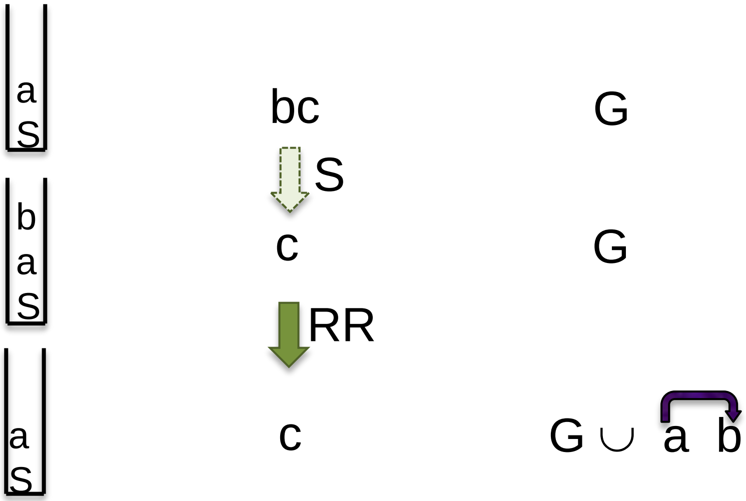


Right-Arc (RA):

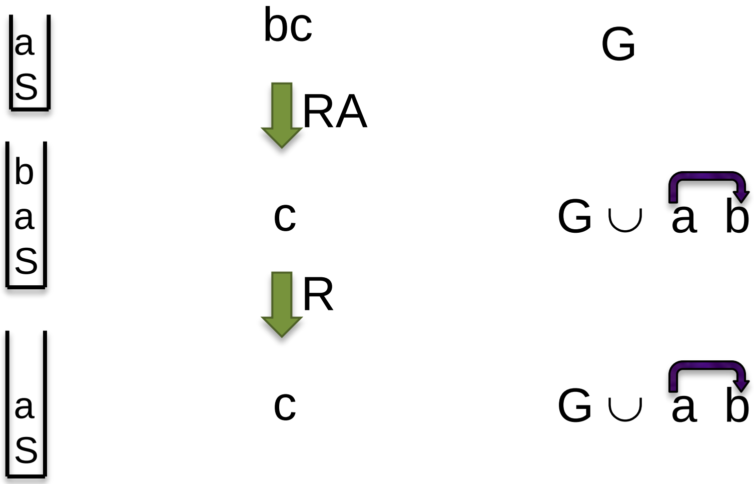


Bottom-up vs. Arc-Eager

Right-Reduce (RR):

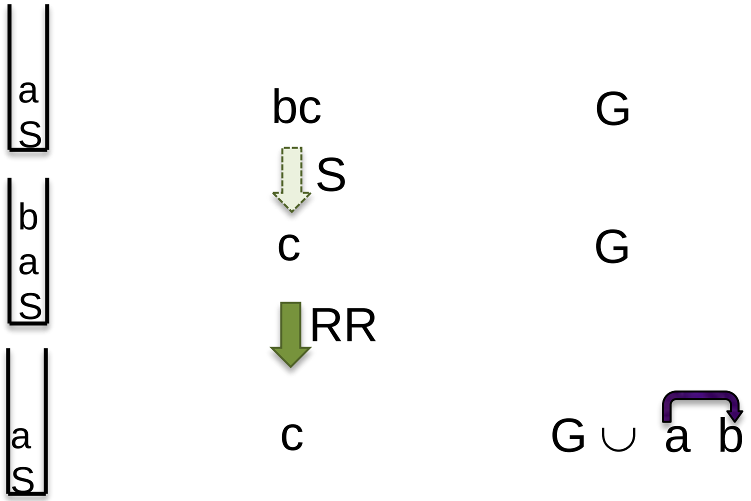


Right-Arc (RA):



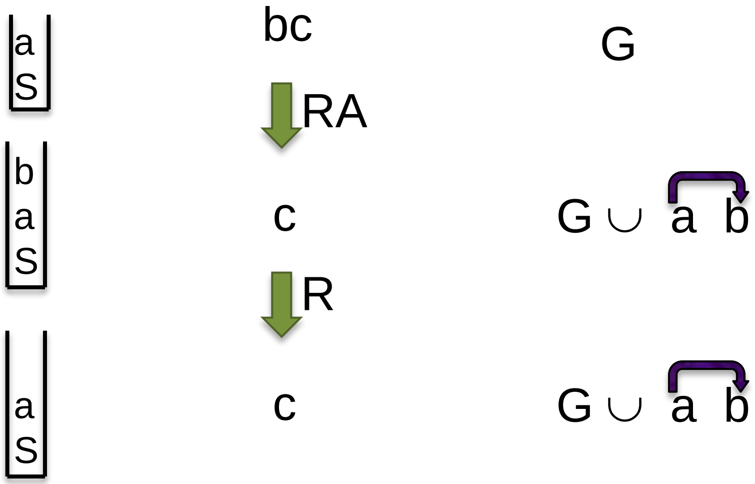
Bottom-up vs. Arc-Eager

Right-Reduce (RR):



a S	bc	G
	S	
b a S	c	G
	RR	
a S	c	G ∪ a b

Right-Arc (RA):

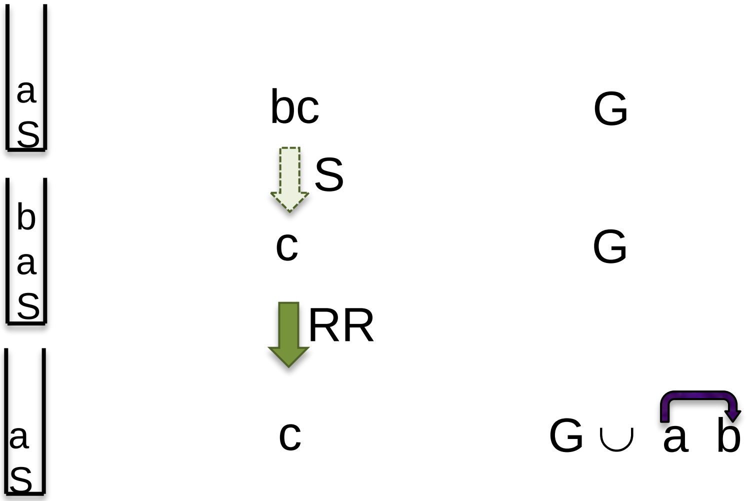


a S	bc	G
	RA	
b a S	c	G ∪ a b
	R	
a S	c	G ∪ a b

→Arc-Eager Dependency Parsing can fully simulate Bottom-up Dependency Parsing!

Bottom-up vs. Arc-Eager

Right-Reduce (RR):



a S | bc | G

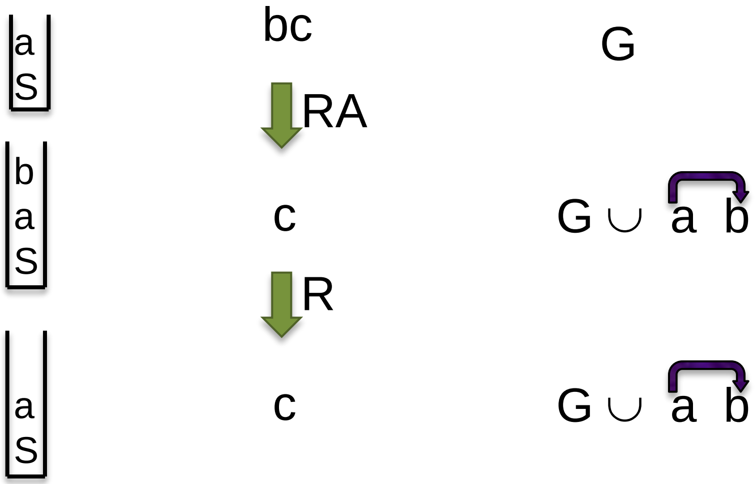
S

b a S | c | G

RR

a S | c | G ∪ a b

Right-Arc (RA):



a S | bc | G

RA

b a S | c | G ∪ a b

R

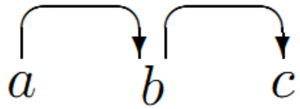
a S | c | G ∪ a b

→Arc-Eager Dependency Parsing can fully simulate Bottom-up Dependency Parsing

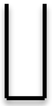
→ We can also derive new graphs with AE! (see next slide)

Arc-Eager Dependency Parsing

(1)



(1) is not derivable with BU parsing, but it is with AE:

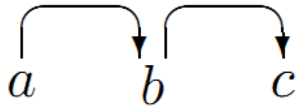


abc

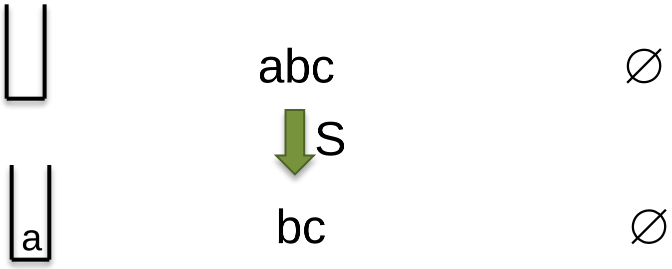
$\varnothing$

Arc-Eager Dependency Parsing

(1)



(1) is not derivable with BU parsing, but it is with AE:

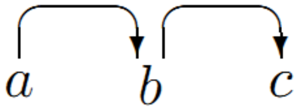


abc ∅

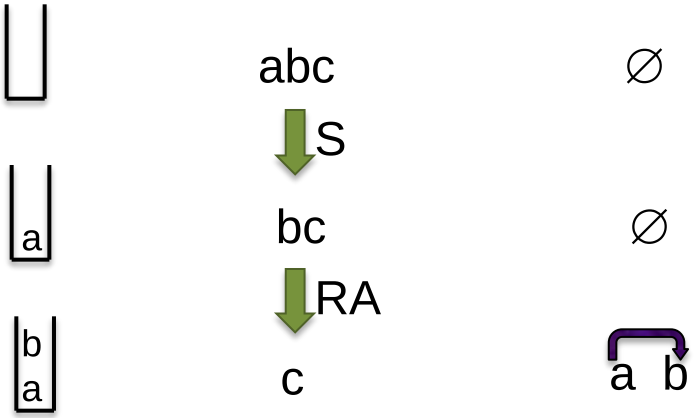
S

a bc ∅

Arc-Eager Dependency Parsing

(1) 

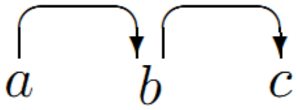
(1) is not derivable with BU parsing, but it is with AE:



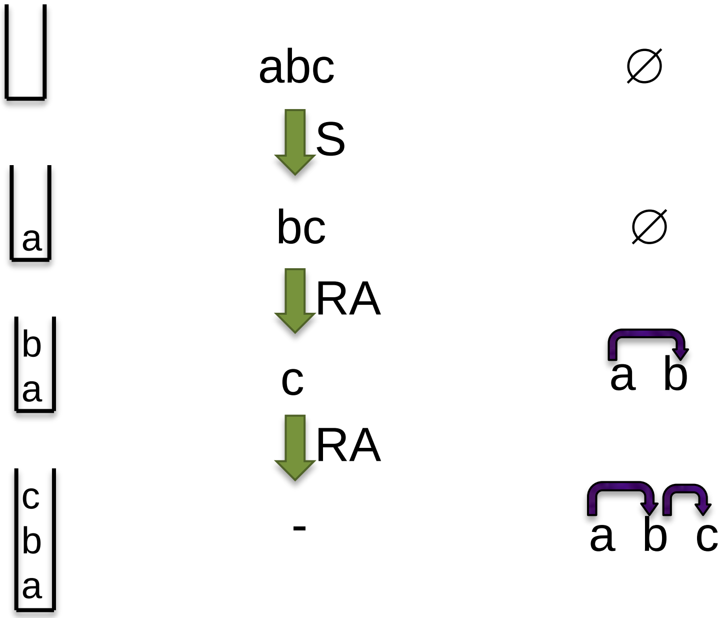
	abc	∅
	S	
a	bc	∅
	RA	
b a	c	a b

Arc-Eager Dependency Parsing

(1)



(1) is not derivable with BU parsing, but it is with AE:

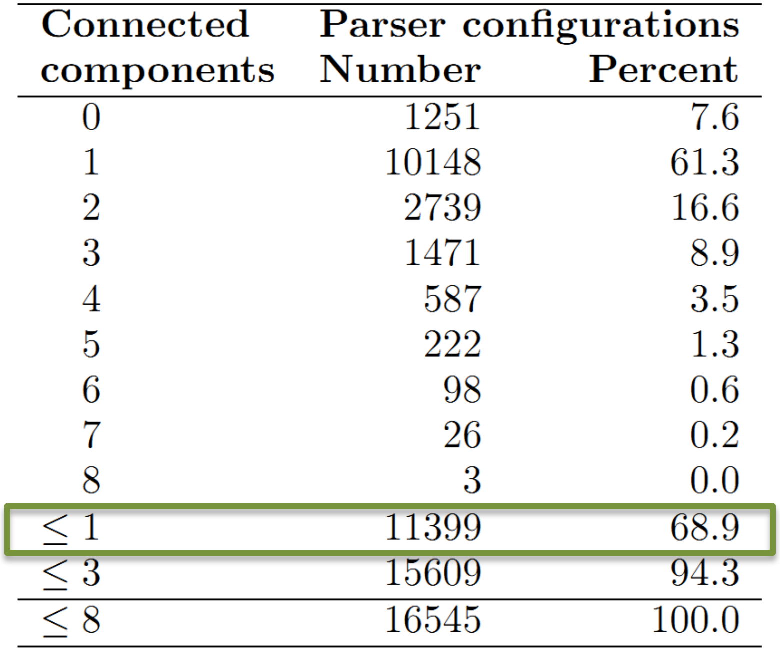


Arc-Eager Parsing: Evaluation

- small Swedish treebank (5685 sentences)
- evaluating incrementality: number of connected components on stack during parse (≤ 1 means strictly incremental)

Arc-Eager Parsing: Evaluation

- small Swedish treebank (5685 sentences)
- evaluating incrementality: number of connected components on stack during parse (≤ 1 means strictly incremental)

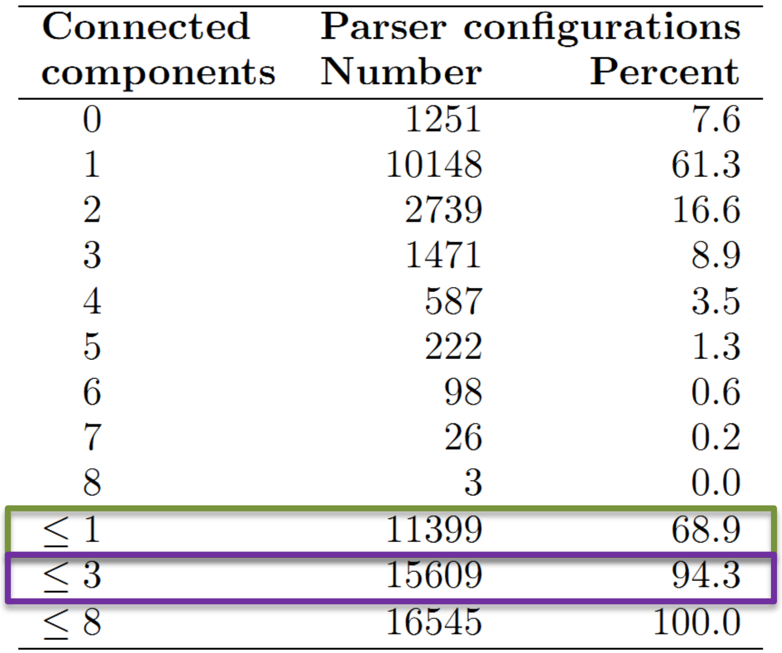


Connected components	Parser configurations	
	Number	Percent
0	1251	7.6
1	10148	61.3
2	2739	16.6
3	1471	8.9
4	587	3.5
5	222	1.3
6	98	0.6
7	26	0.2
8	3	0.0
≤ 1	11399	68.9
≤ 3	15609	94.3
≤ 8	16545	100.0

strictly incremental

Arc-Eager Parsing: Evaluation

- small Swedish treebank (5685 sentences)
- evaluating incrementality: number of connected components on stack during parse (≤ 1 means strictly incremental)



Connected	Parser configurations	
components	Number	Percent
0	1251	7.6
1	10148	61.3
2	2739	16.6
3	1471	8.9
4	587	3.5
5	222	1.3
6	98	0.6
7	26	0.2
8	3	0.0
≤ 1	11399	68.9
≤ 3	15609	94.3
≤ 8	16545	100.0

strictly incremental

“mildly” incremental

Intermediate Summary

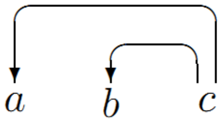
- Dependency parsing works well in practice
- Incremental dependency parsing possible in many cases
- Improving the parsing technique is essential
  - Arc-Eager performs better than Bottom-up dep. parsing
  - Well-formed parsing results show high incrementality

Intermediate Summary

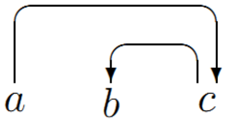
- Dependency parsing works well in practice
- Incremental dependency parsing possible in many cases
- Improving the parsing technique is essential
  - Arc-Eager performs better than Bottom-up dep. parsing
  - Well-formed parsing results show high incrementality

- … but, what about those structures (6) and (7) we couldn’t parse incrementally?

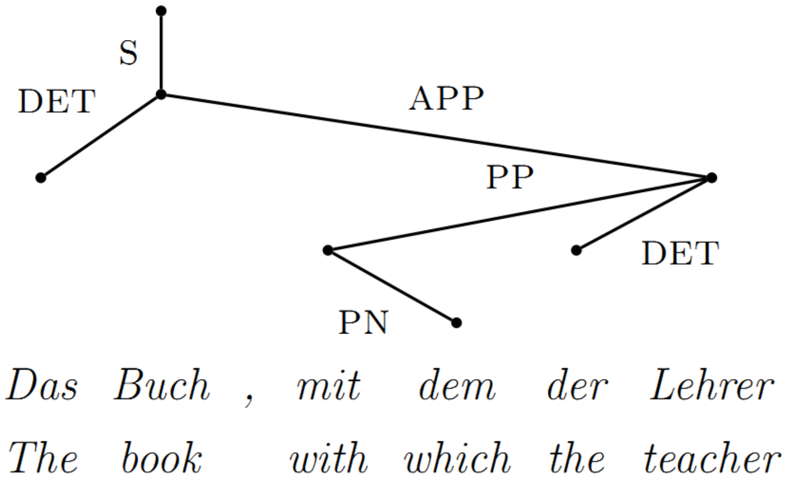
(6)



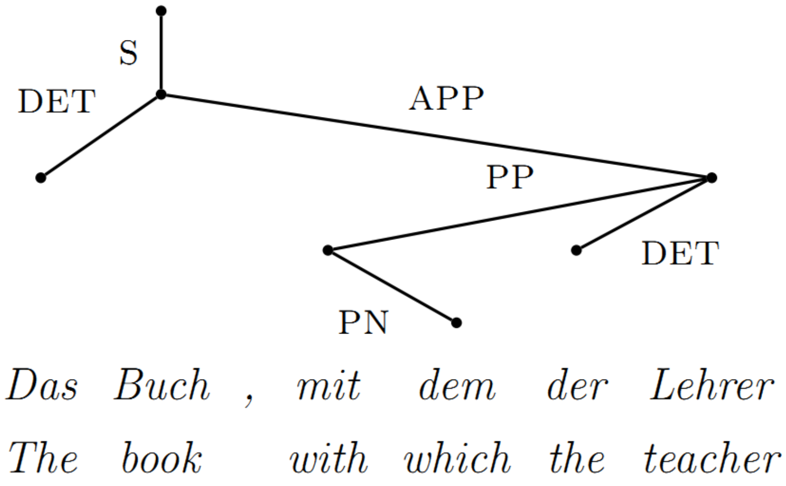
(7)



Robust Incrementality



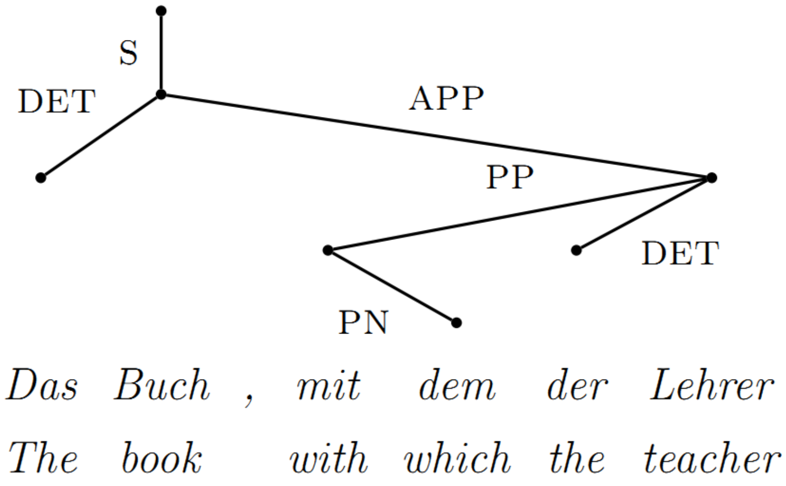
Robust Incrementality



Drawbacks of storing components on a stack

- psycholinguistic plausibility: why not integrate directly?

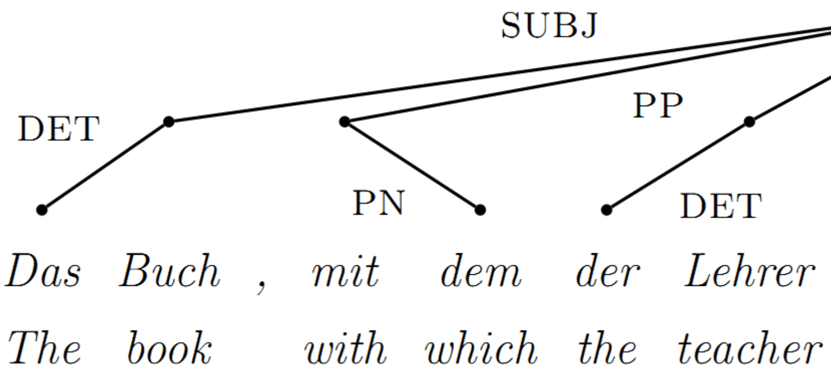
Robust Incrementality



Drawbacks of storing components on a stack

- psycholinguistic plausibility: why not integrate directly?
- practicality: delay of output as stored components are not part of it

Robust Incrementality



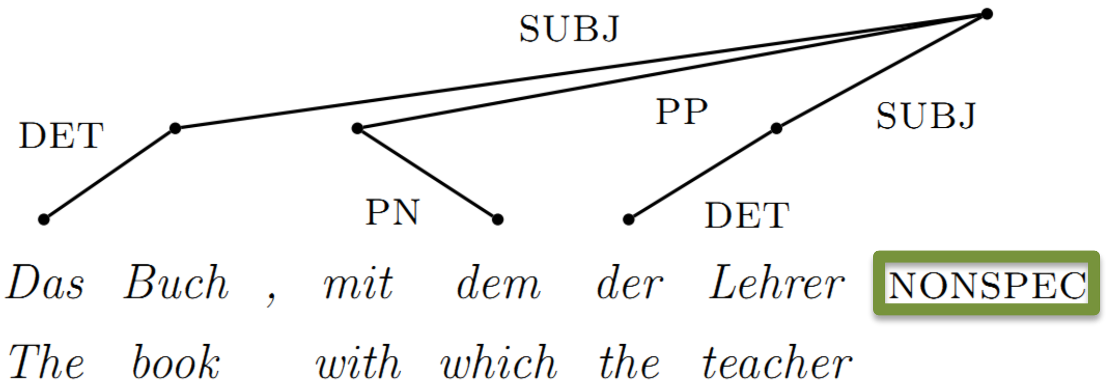
- Argument Dependency Model
  - dependencies between a verb’s arguments
  - proto roles (proto-agent, proto-patient)
    - e.g.: noun(animate & nominative)

      → noun(proto-agent)

      → dependency rel. SUBJ governs the noun (phrase)

      → unless contradictory constraints override this

Robust Incrementality



NONSPEC node

- connect structures to NONSPEC node while verb has not been found
- NONSPEC can change into any other node and even divide into several nodes
- May even be in the resulting graph

Robust Incrementality: Evaluation

- corpus with
  - uniform sentence pattern
  - verb-final sublauses

→ 97.3% accurate dependency graphs, but…

Summary

- Incremental Dependency Parsing is possible and efficient
- Verb-end structures pose problems to strict incrementality

- Pseudo-strict incrementality with abstract NONSPEC node suggested
- Integrates dep. relations on-the-fly
- still seems a lot like a renamed stack to me (which can be output) → too vague

Thank you!

References

- Joakim Nivre (2004). Incrementality in Deterministic Dependency Parsing
- Wolfgang Menzel (2009). Towards radically incremental parsing of natural language