Filtering Axiom Links

Richard Moot

LaBRI CNRS and INRIA Futurs

Formal Grammar, 4–5 August 2007

Outline

Outline

Outline

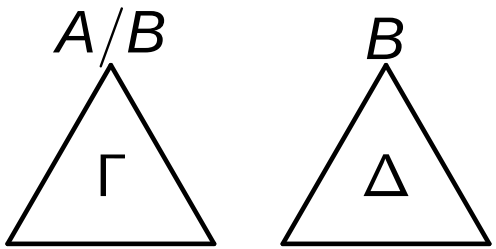
Outline

Outline

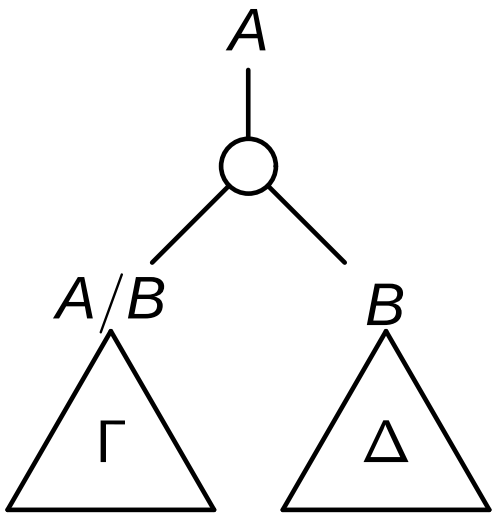
Introduction

- The Non-associative Lambek calculus (NL) was introduced by Lambek in 1961.
- We will look at *extending* NL to take into account more than just contex free languages.
- I will show how to use several simple mechanisms to prevent partial parses which can never be completed.
- I will evaluate the effectiveness of these strategies on a database of randomly generated Lambek calculus proofs.

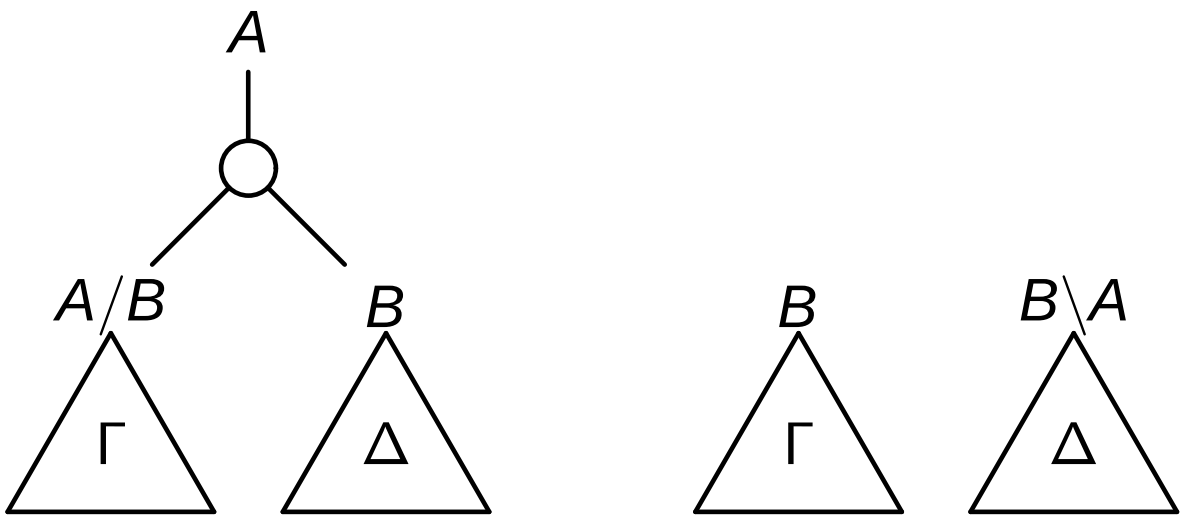
AB Grammars
Logical Rules



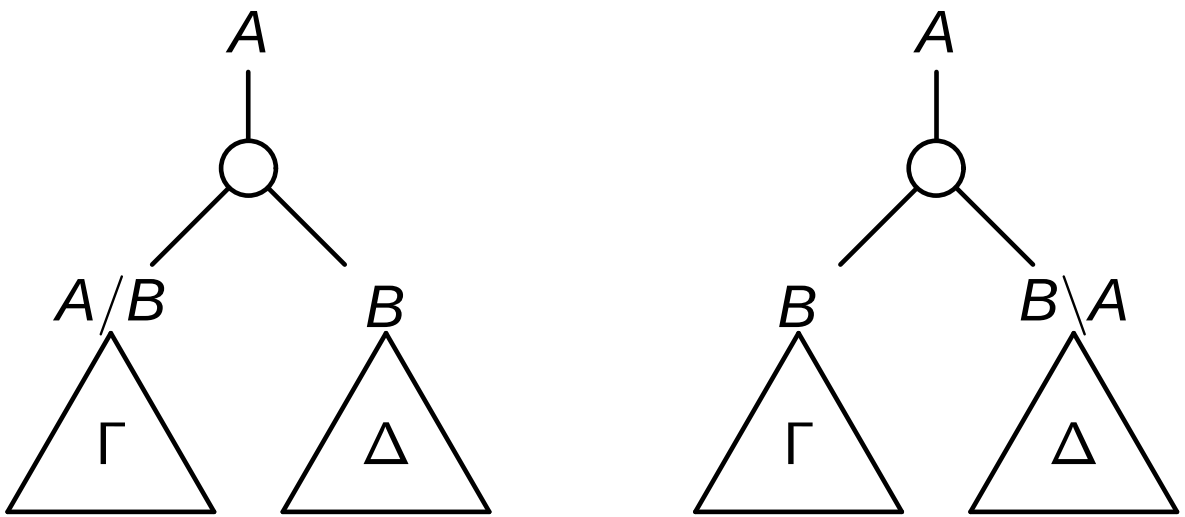
AB Grammars
Logical Rules



AB Grammars
Logical Rules



AB Grammars
Logical Rules



AB Grammars

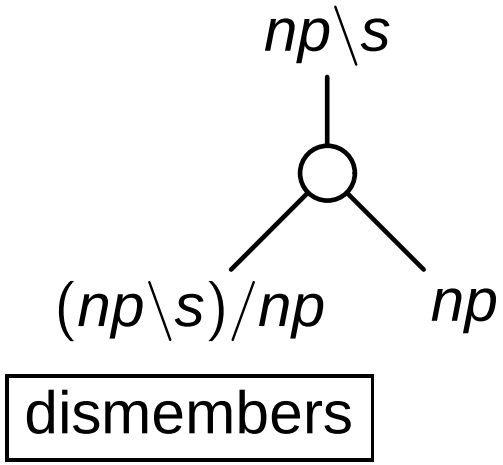
Compiling Away The Lexicon

$$(np\backslash s)/np$$

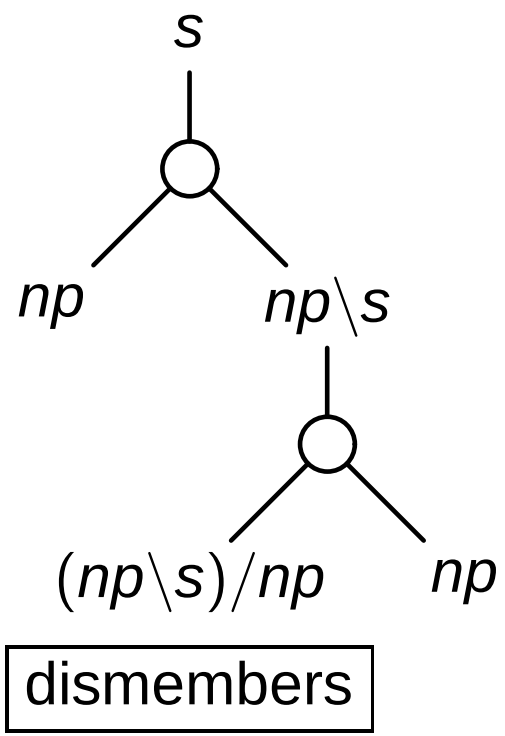
dismembers

AB Grammars

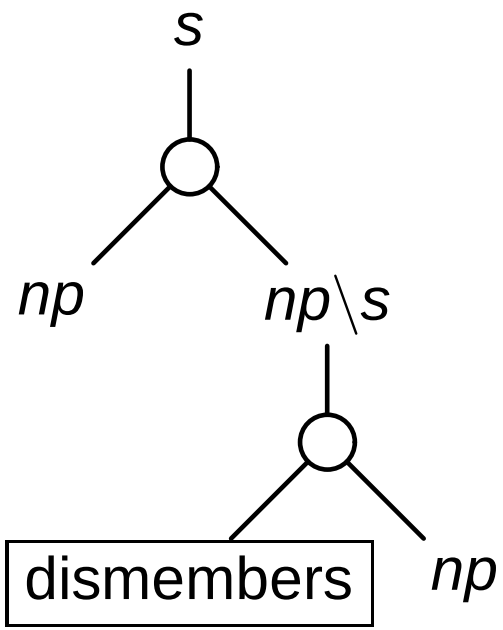
Compiling Away The Lexicon



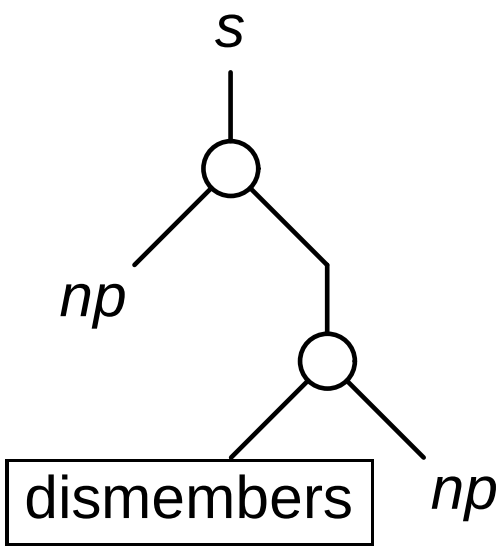
AB Grammars
Compiling Away The Lexicon



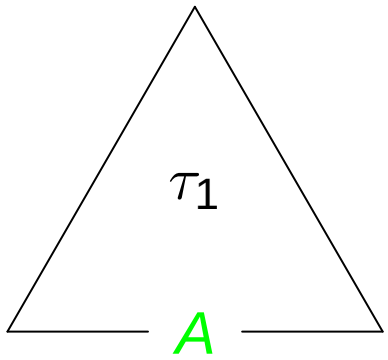
AB Grammars
Compiling Away The Lexicon



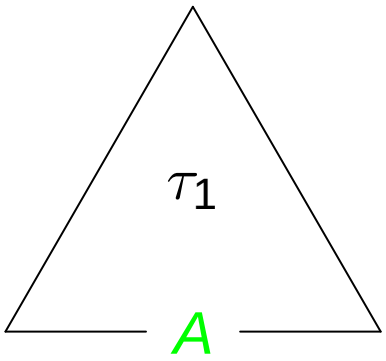
AB Grammars
Compiling Away The Lexicon

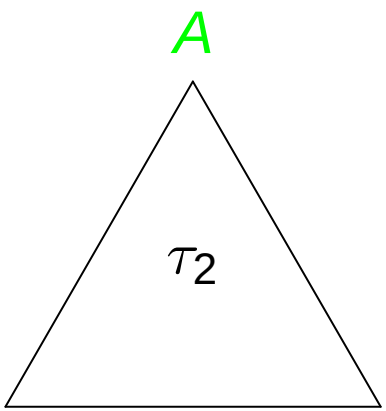


AB Grammars
The Axiom Rule

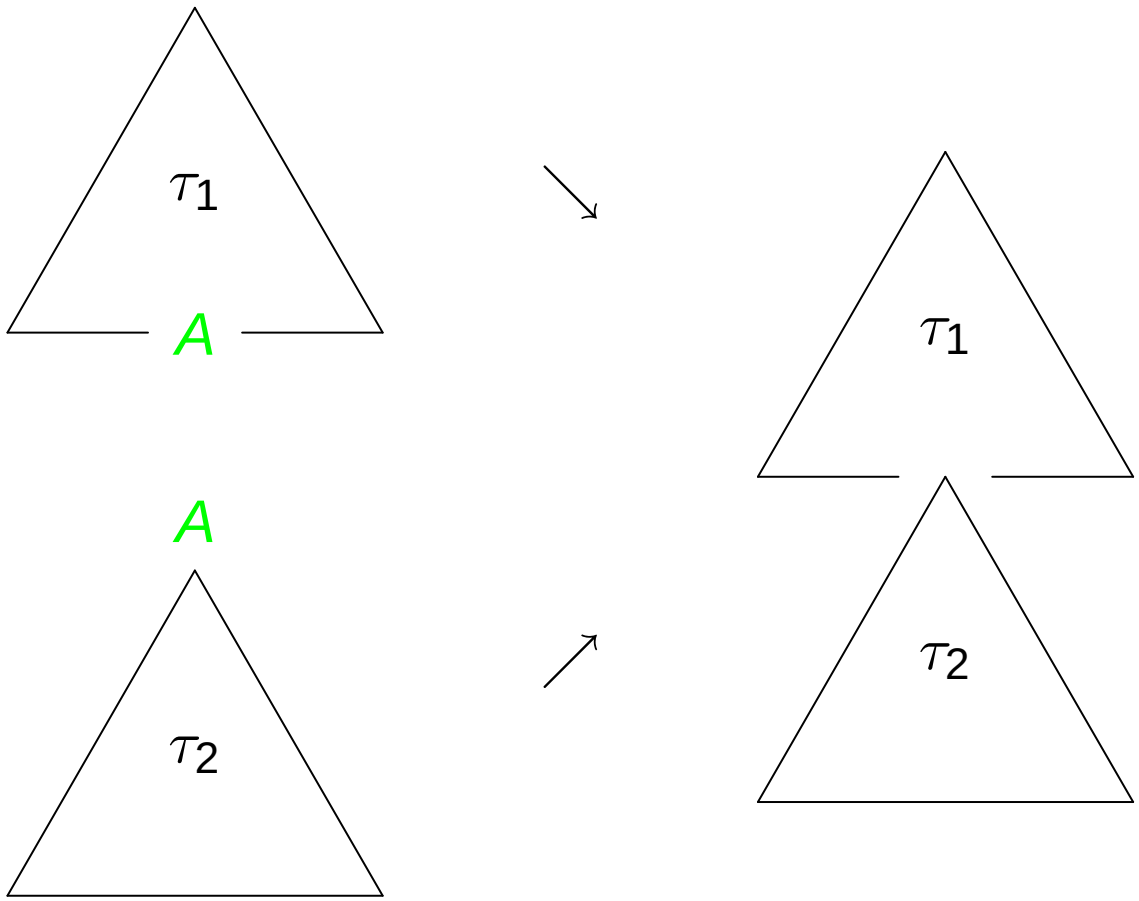


AB Grammars
The Axiom Rule



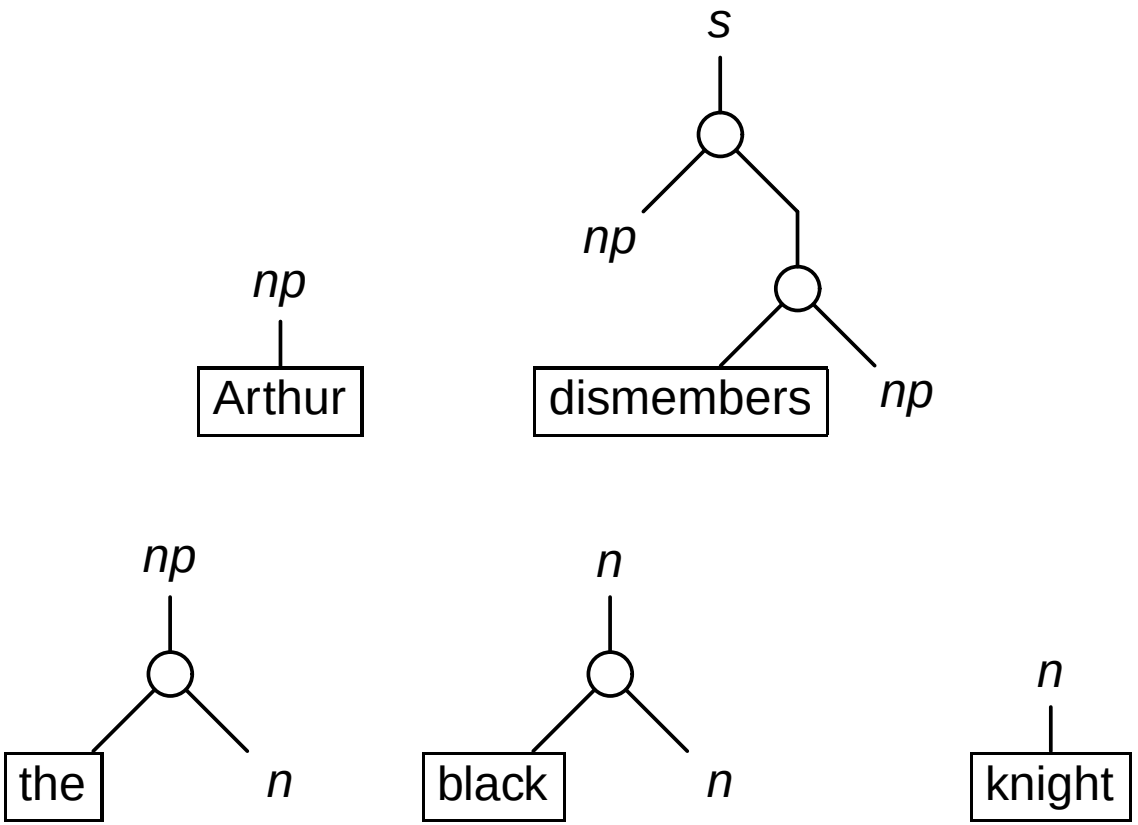


AB Grammars
The Axiom Rule



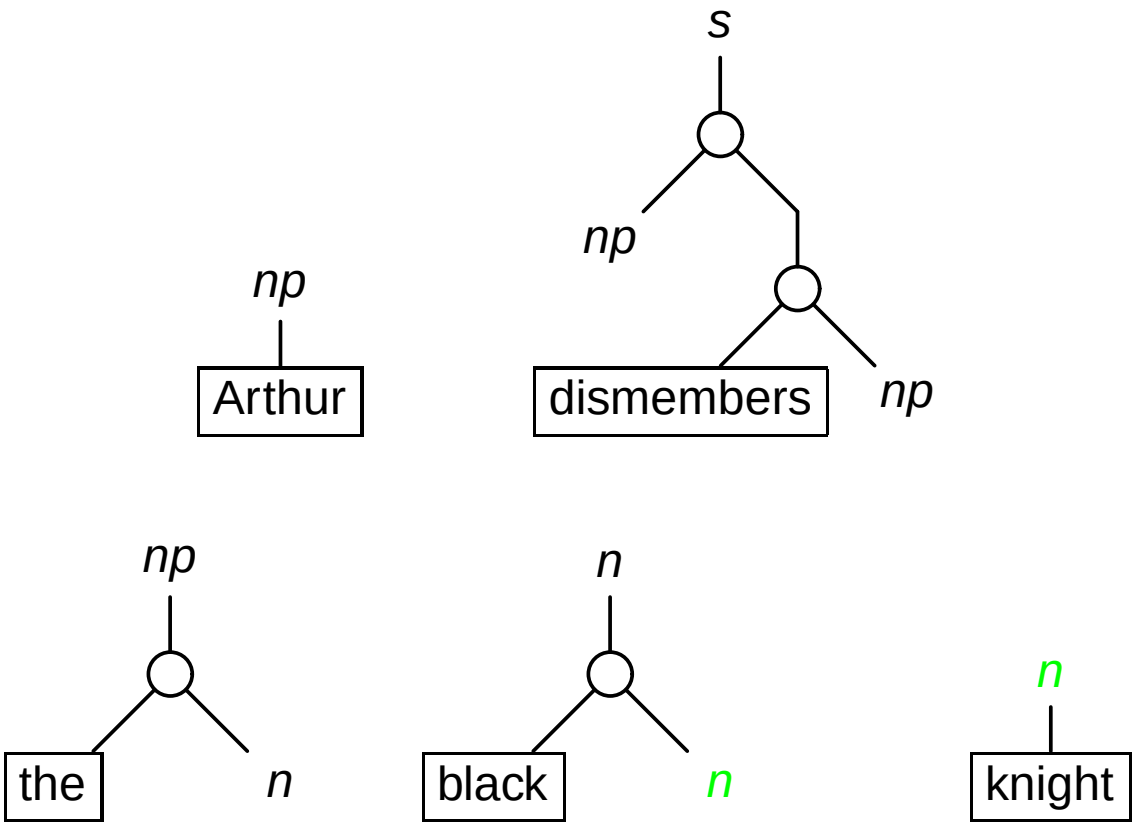
AB Grammars

Example: ‘Arthur dismembers the black knight’



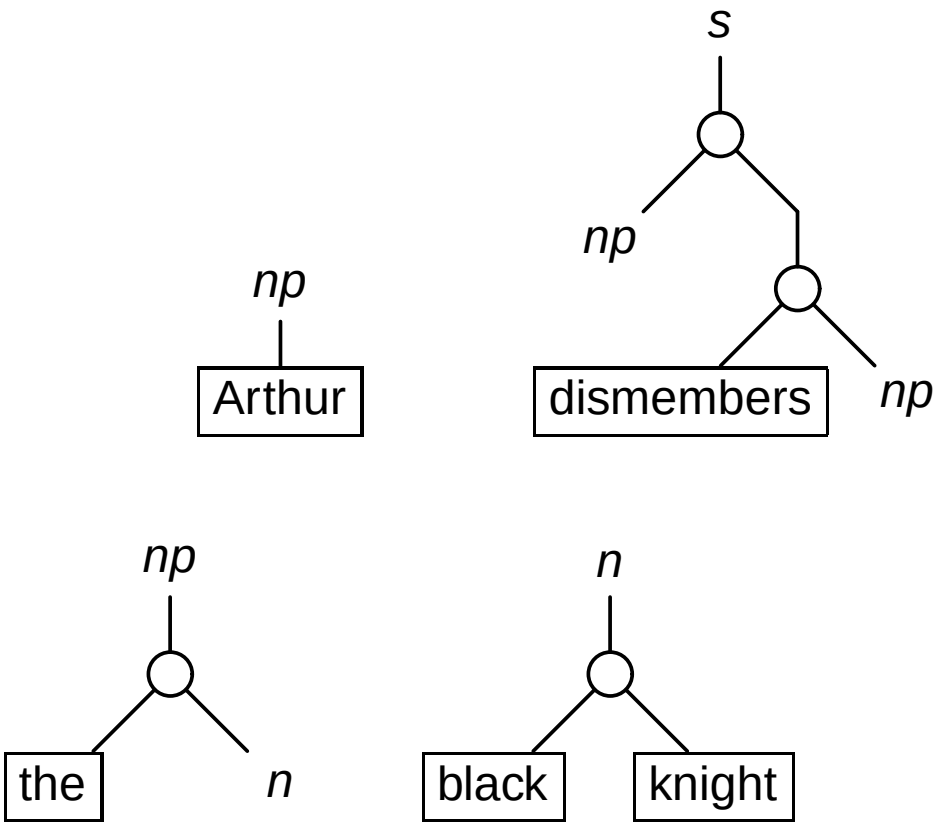
AB Grammars

Example: ‘Arthur dismembers the black knight’



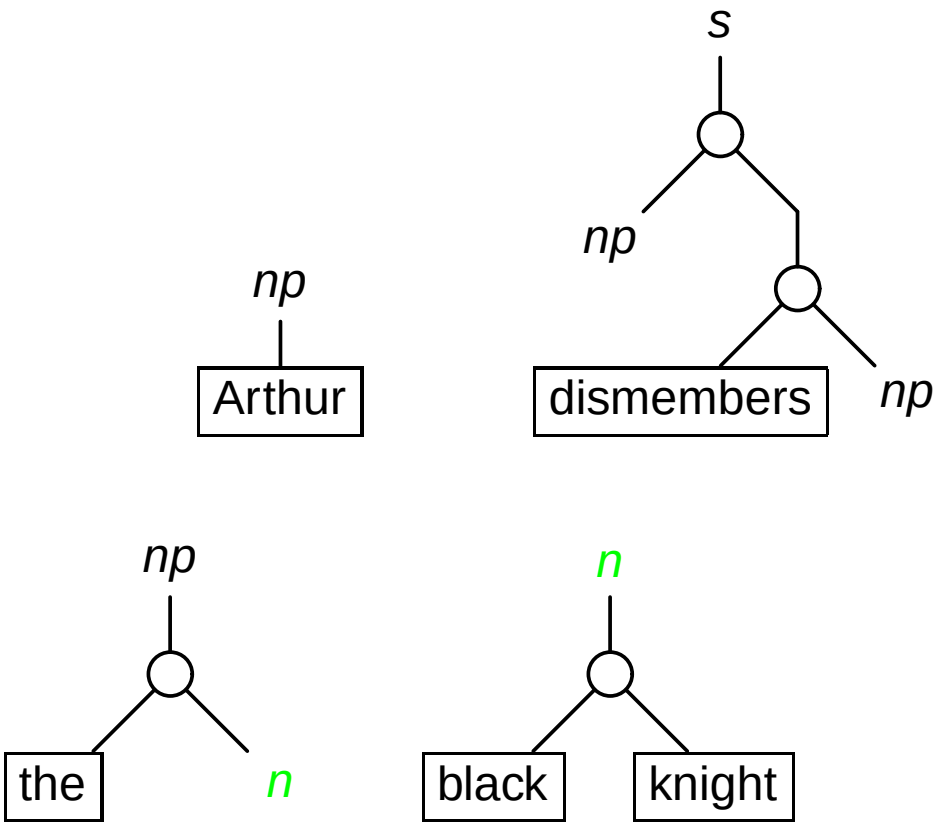
AB Grammars

Example: ‘Arthur dismembers the black knight’



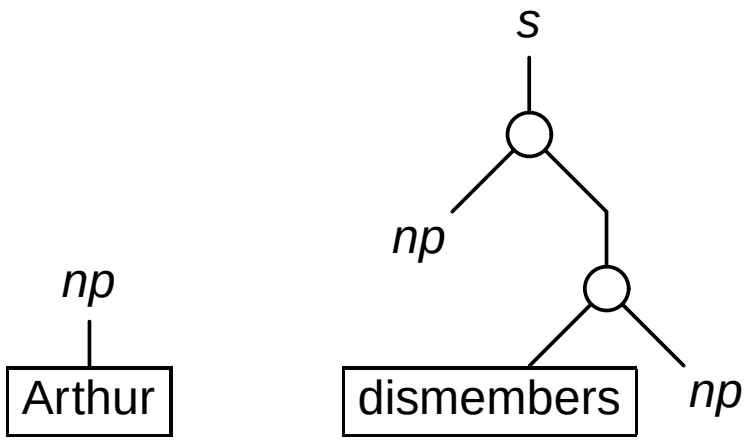
AB Grammars

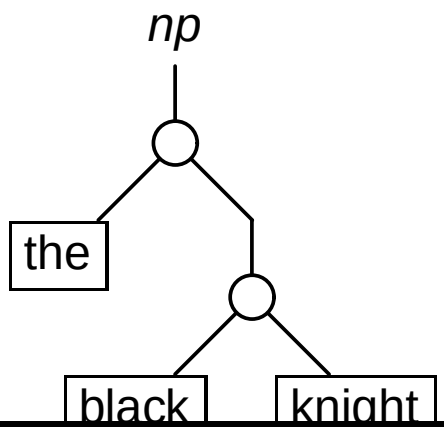
Example: ‘Arthur dismembers the black knight’



AB Grammars

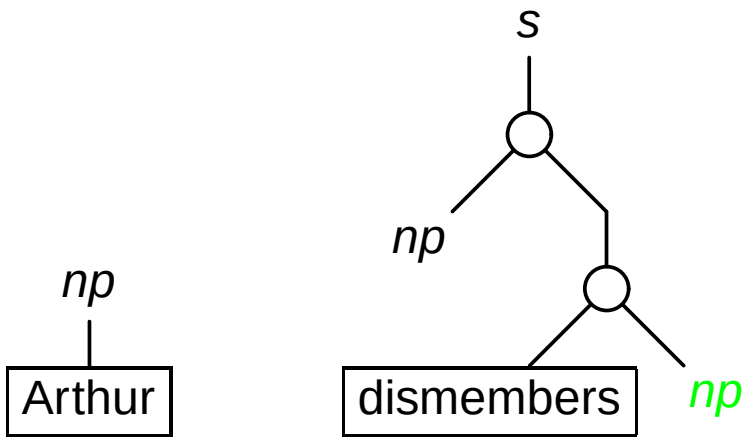
Example: ‘Arthur dismembers the black knight’





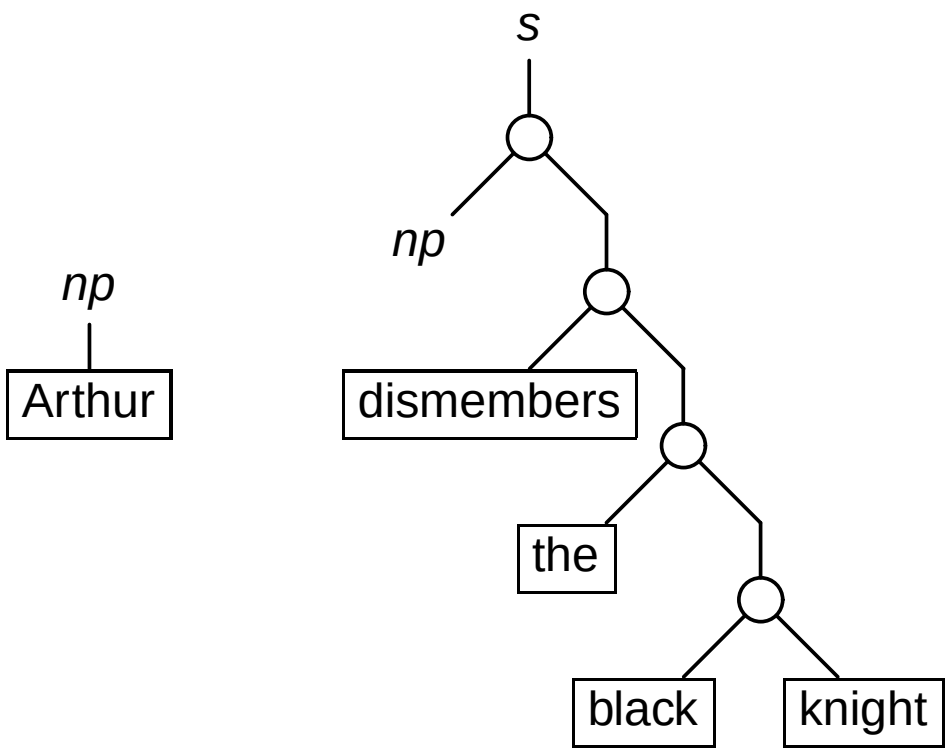
AB Grammars

Example: ‘Arthur dismembers the black knight’



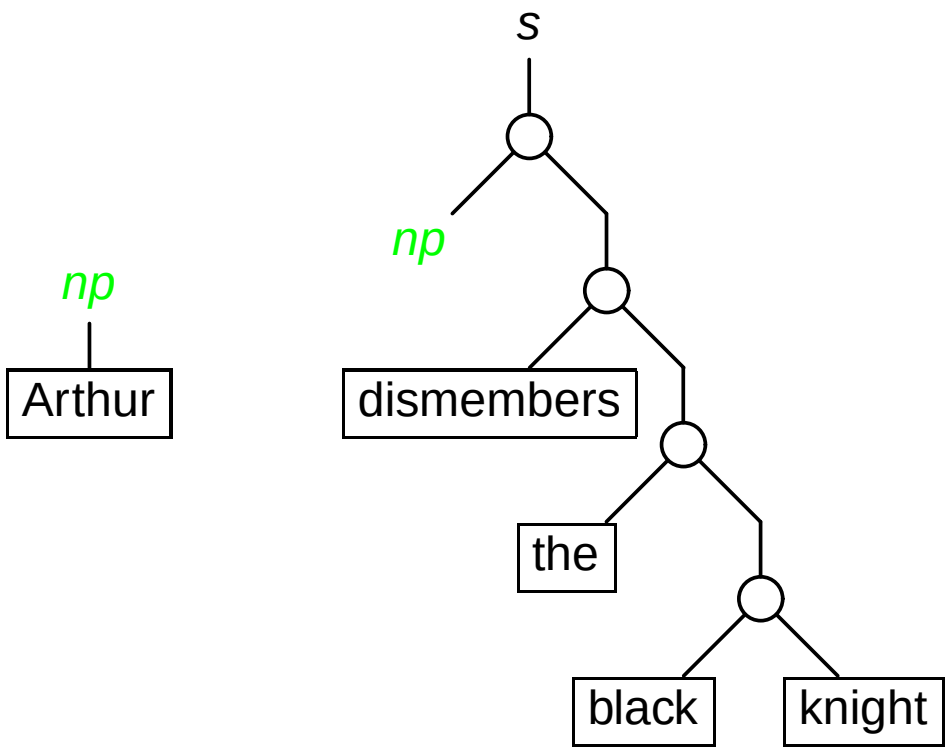
AB Grammars

Example: ‘Arthur dismembers the black knight’



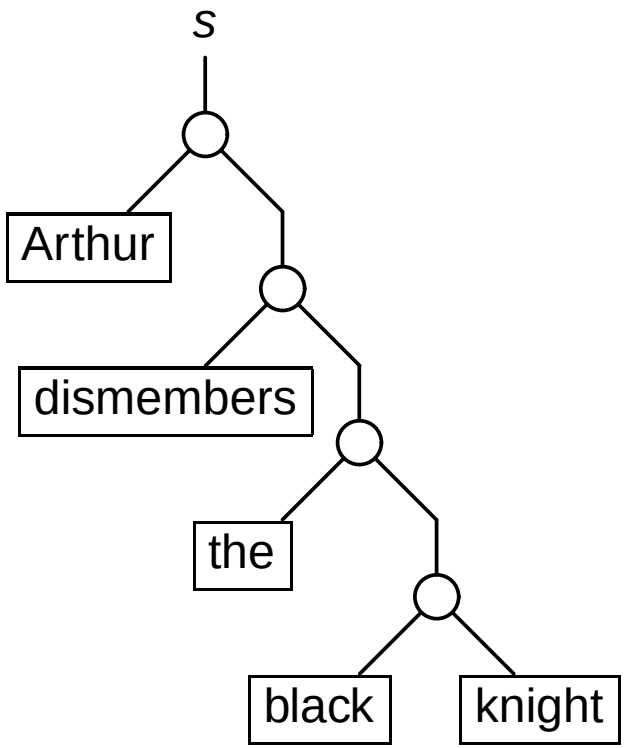
AB Grammars

Example: ‘Arthur dismembers the black knight’

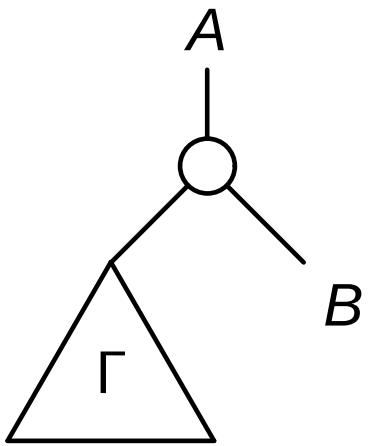


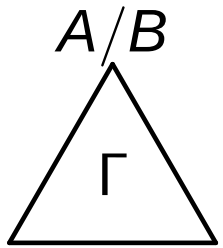
AB Grammars

Example: ‘Arthur dismembers the black knight’

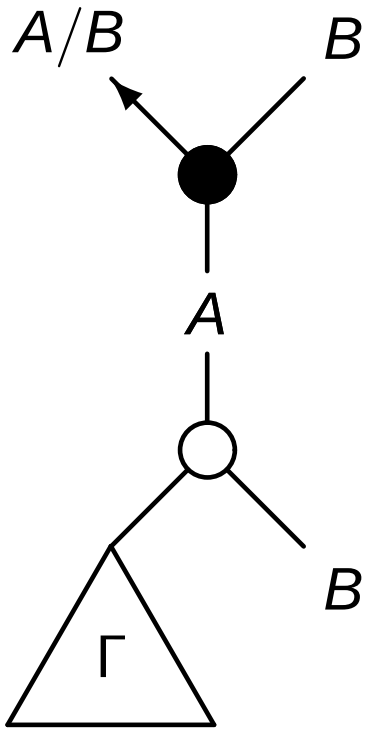


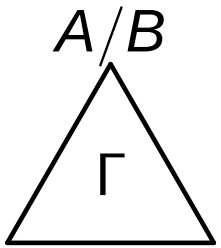
NL Grammars
Logical Rules



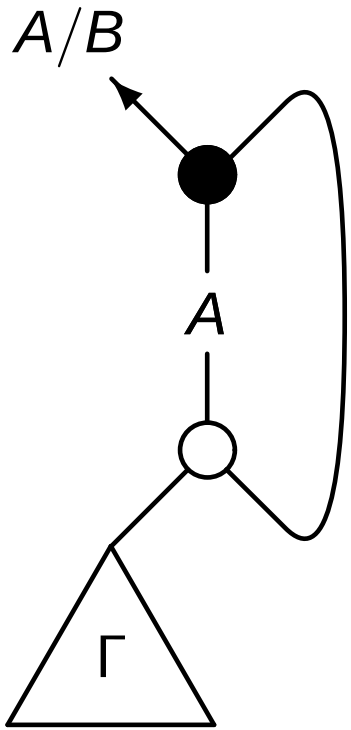


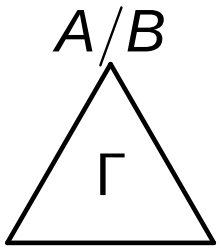
NL Grammars
Logical Rules



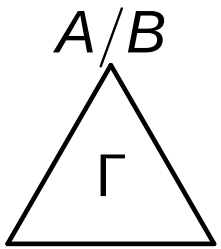


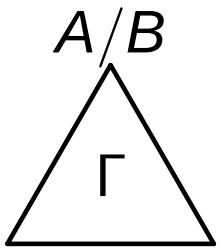
NL Grammars
Logical Rules



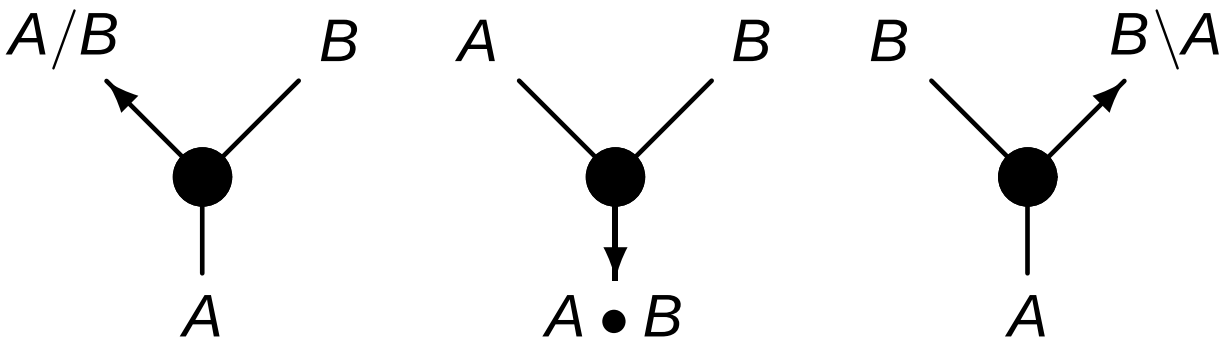


NL Grammars
Logical Rules

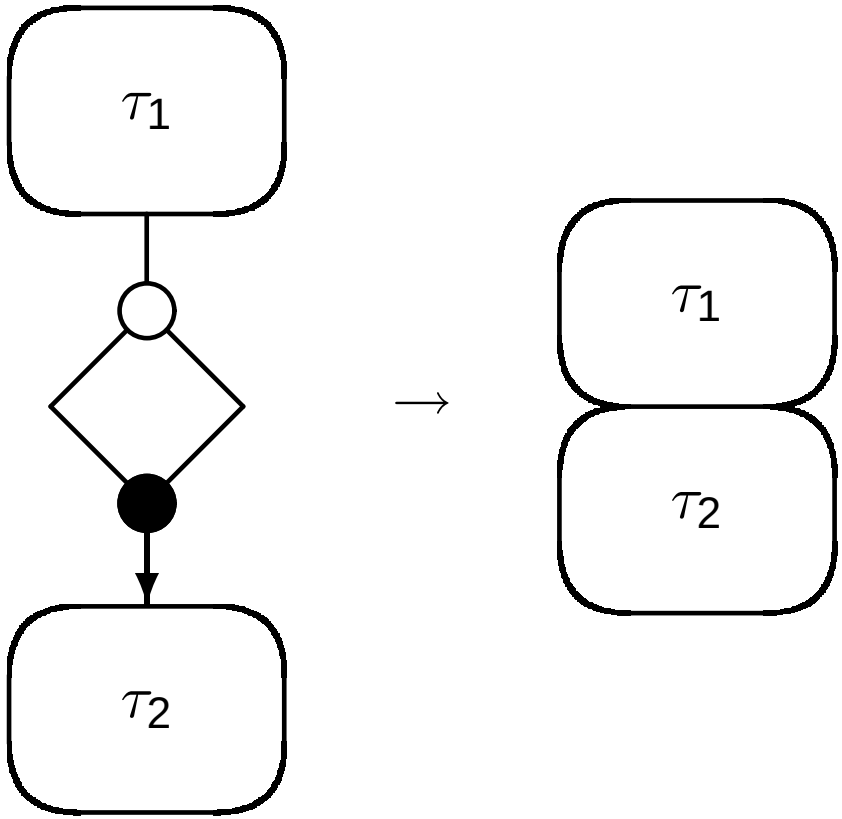




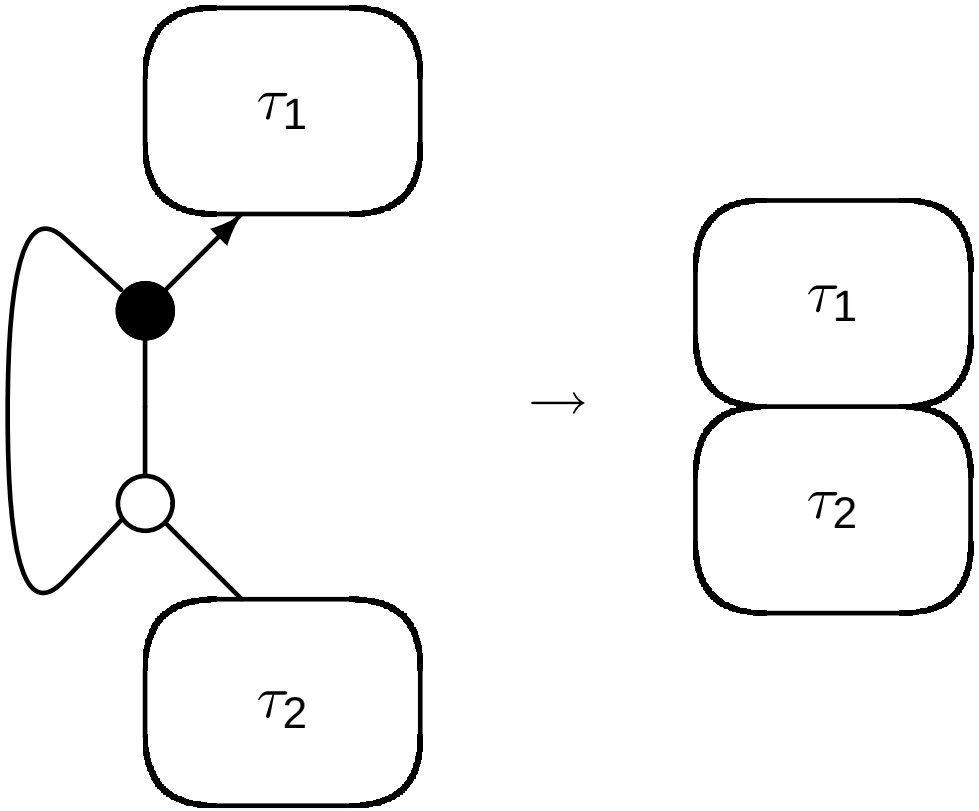
NL Grammars
Links



NL Grammars
Graph Contractions

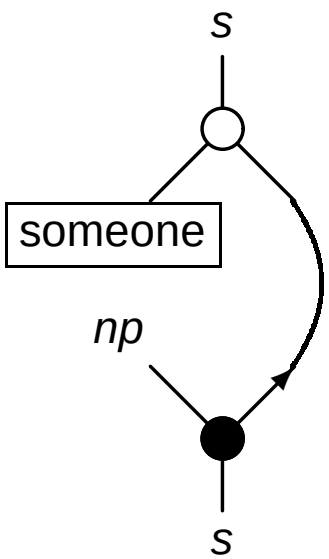


NL Grammars
Graph Contractions



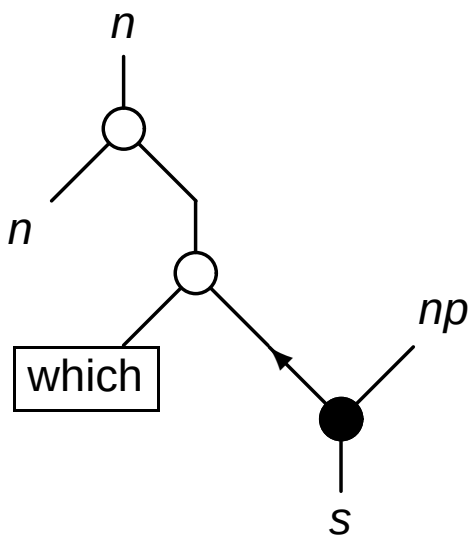
NL Grammars

Example: Quantifiers



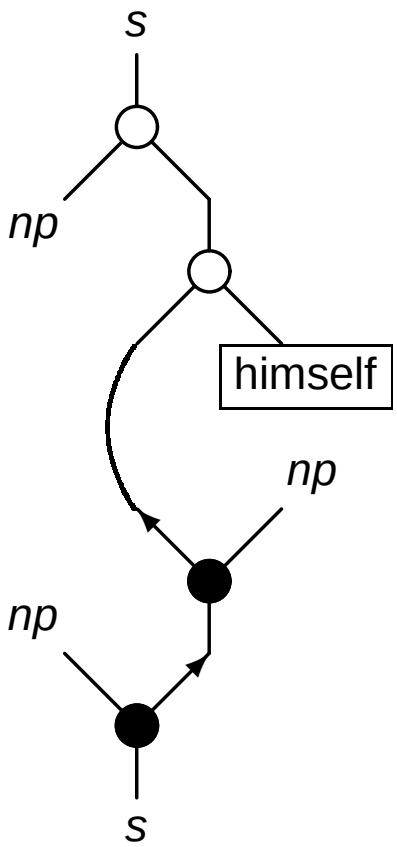
NL Grammars

Example: Extraction



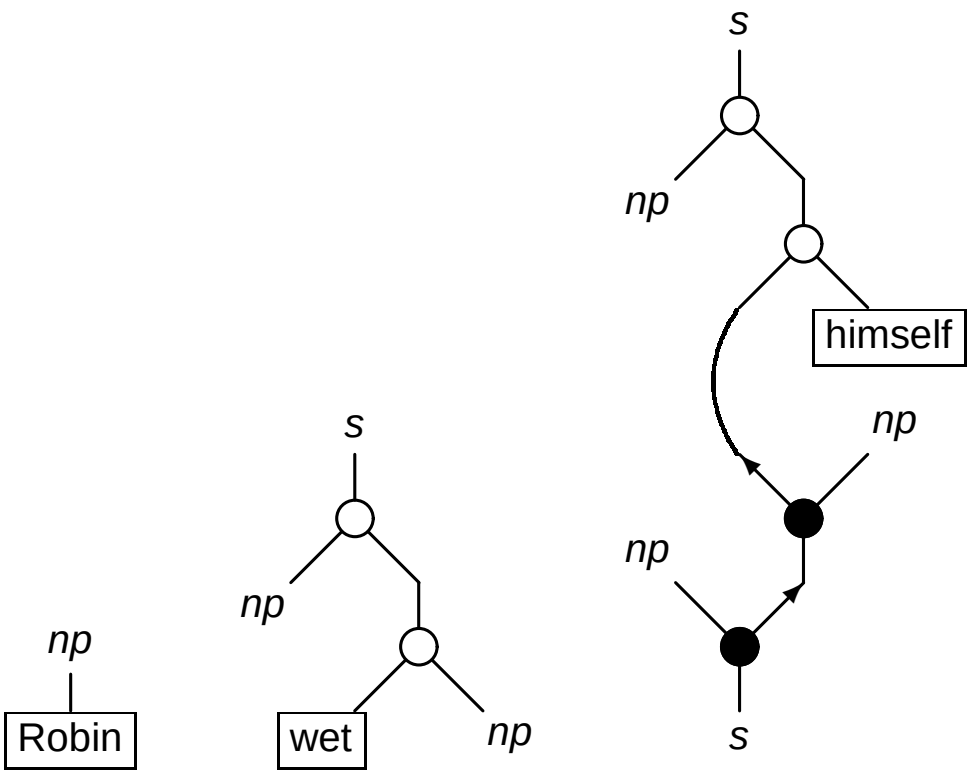
NL Grammars

Example: Reflexives



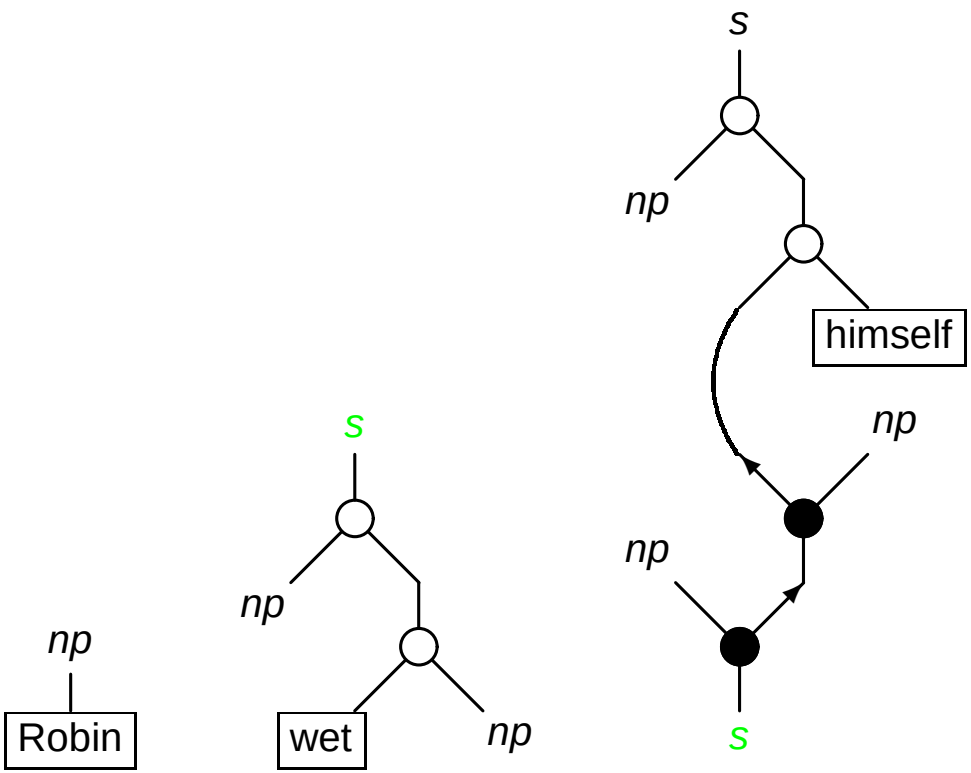
NL Grammars

Example: ‘Robin wet himself’



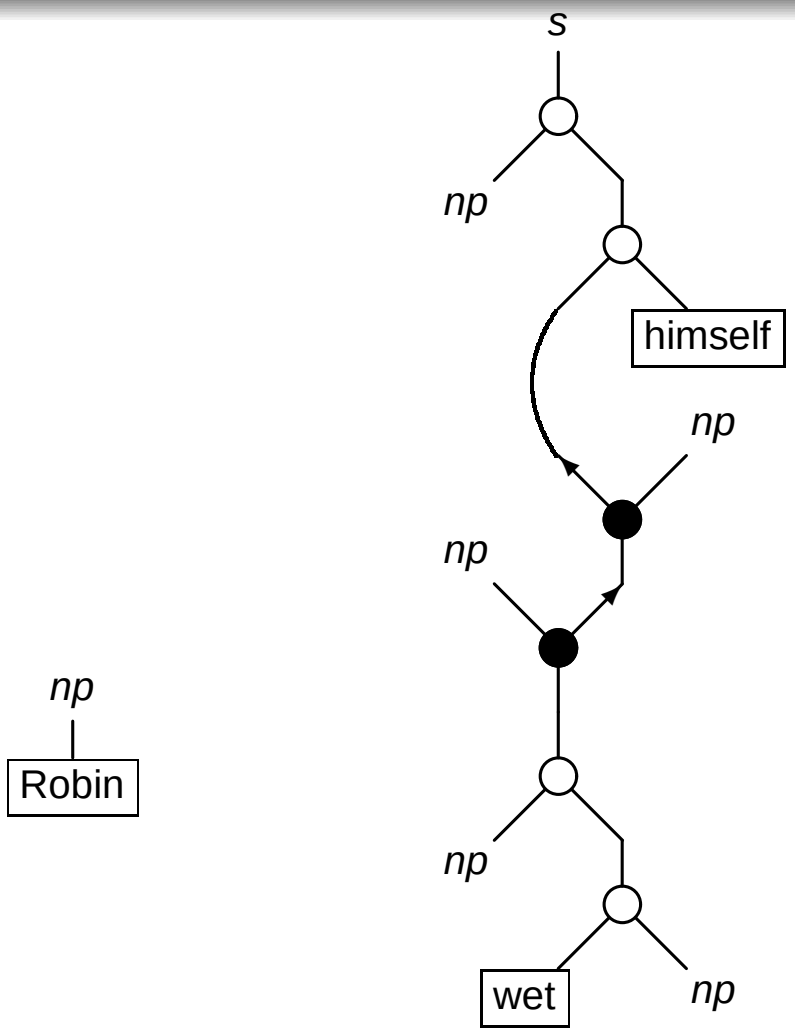
NL Grammars

Example: ‘Robin wet himself’



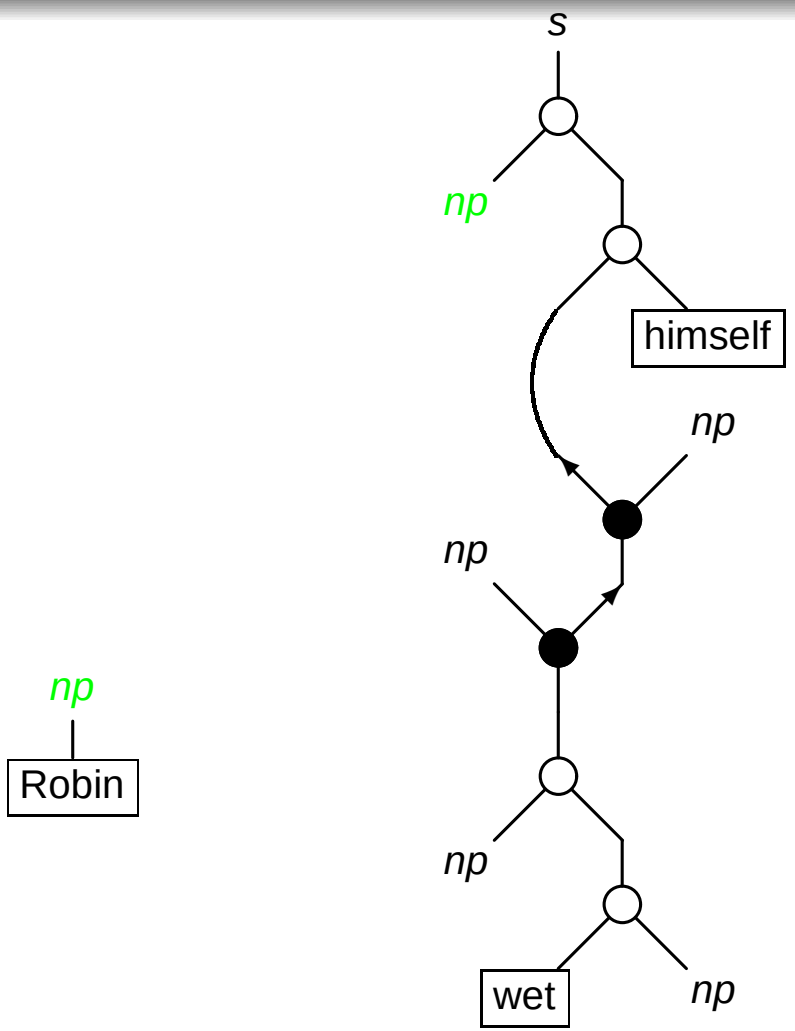
NL Grammars

Example: ‘Robin wet himself’



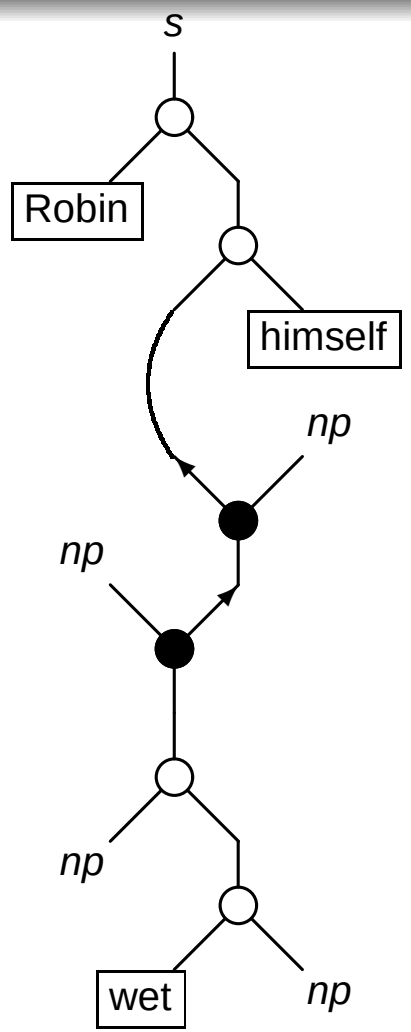
NL Grammars

Example: ‘Robin wet himself’



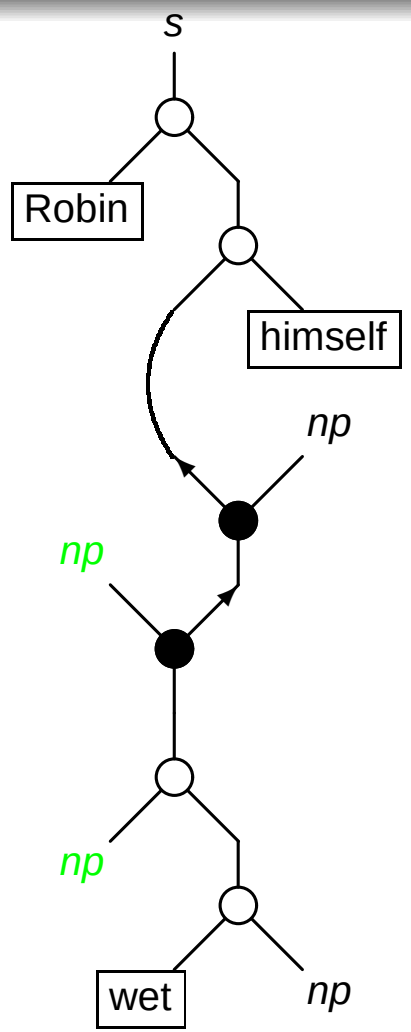
NL Grammars

Example: ‘Robin wet himself’



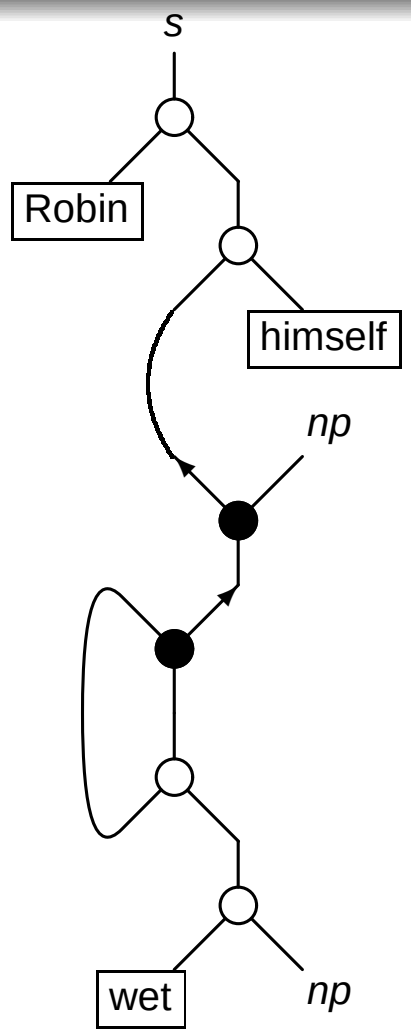
NL Grammars

Example: ‘Robin wet himself’



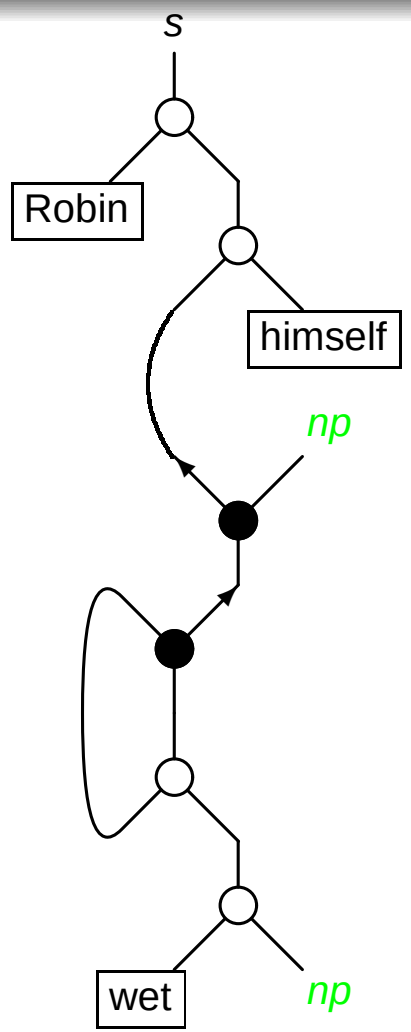
NL Grammars

Example: ‘Robin wet himself’



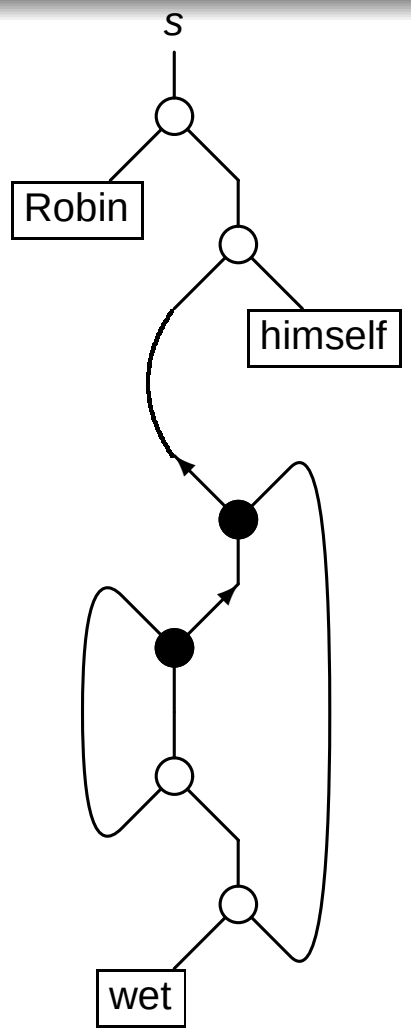
NL Grammars

Example: ‘Robin wet himself’



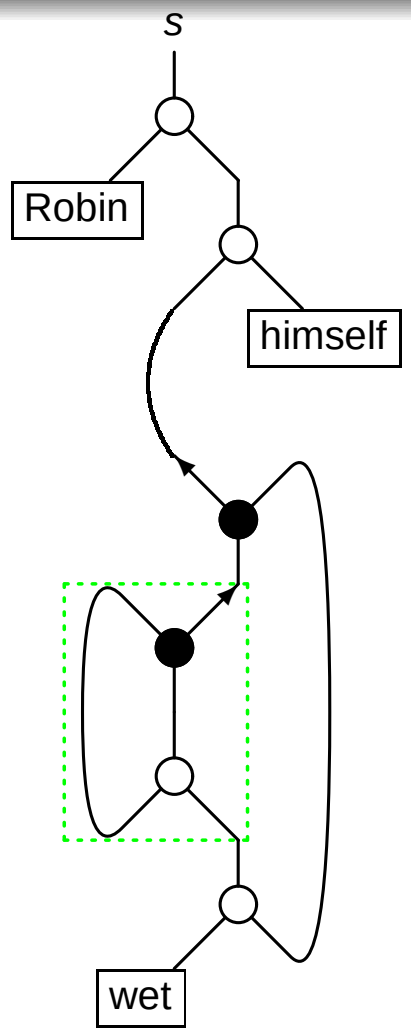
NL Grammars

Example: ‘Robin wet himself’



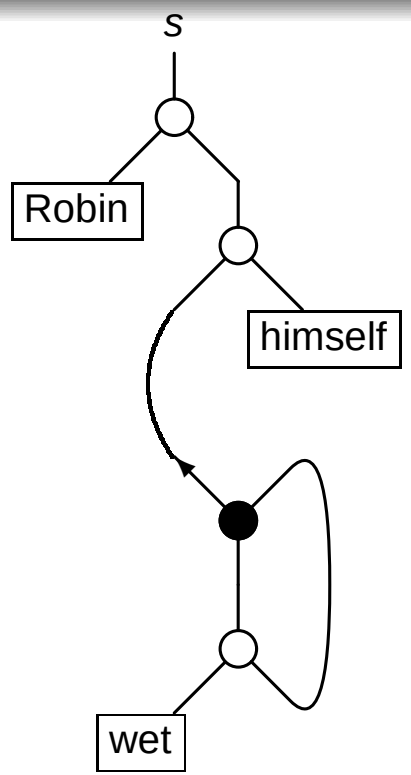
NL Grammars

Example: ‘Robin wet himself’



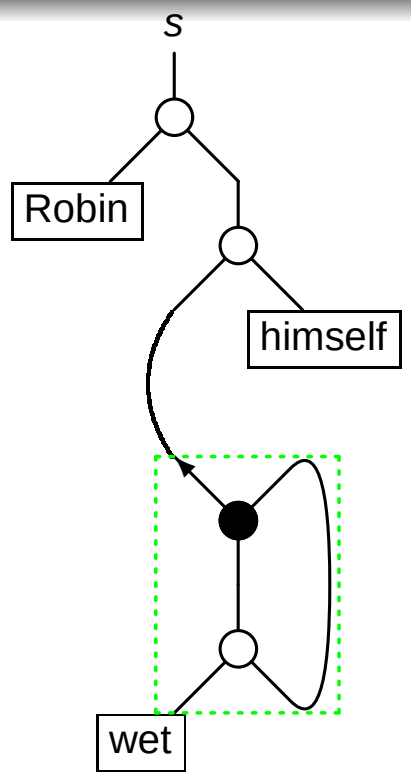
NL Grammars

Example: ‘Robin wet himself’



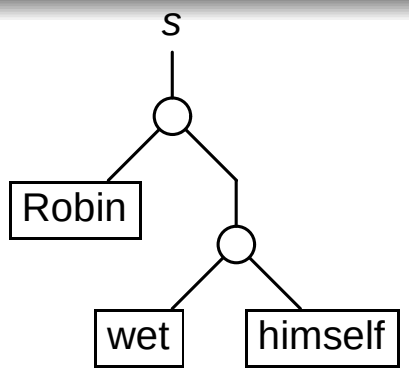
NL Grammars

Example: ‘Robin wet himself’



NL Grammars

Example: ‘Robin wet himself’

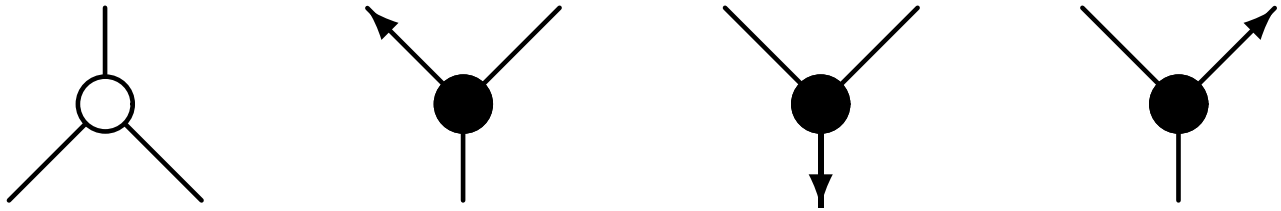


Multimodal Categorial Grammars

- NL is an interesting formalism, but it recognizes only context free languages.
- To boost its expressive power, as well as increase the level of detail in the analyses, we will look at extensions of NL.

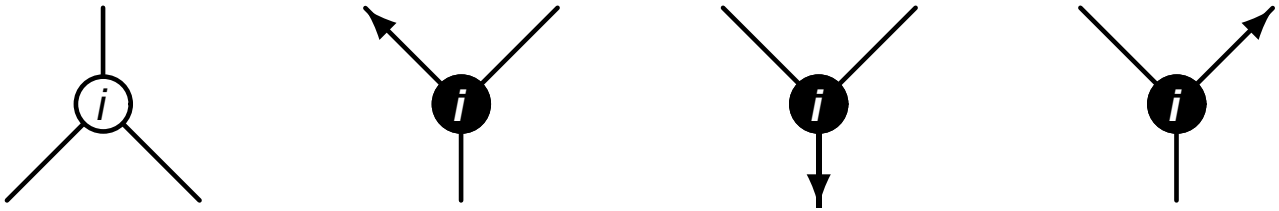
Multimodal Categorial Grammars

Mode Information



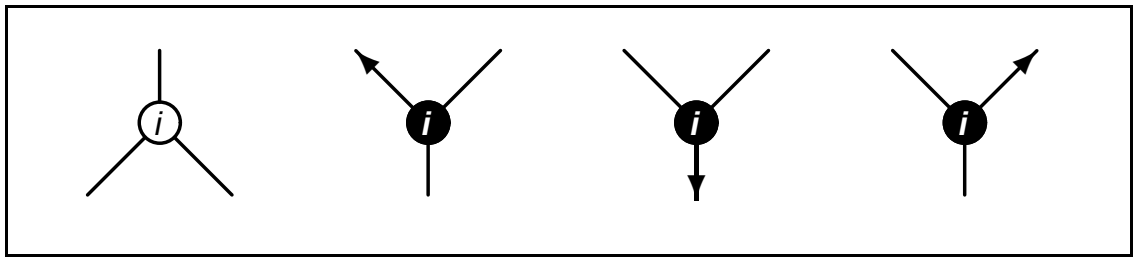
Multimodal Categorial Grammars

Mode Information



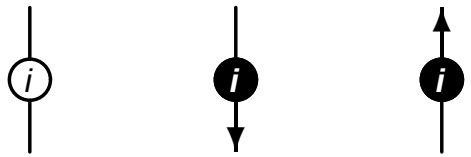
Multimodal Categorial Grammars

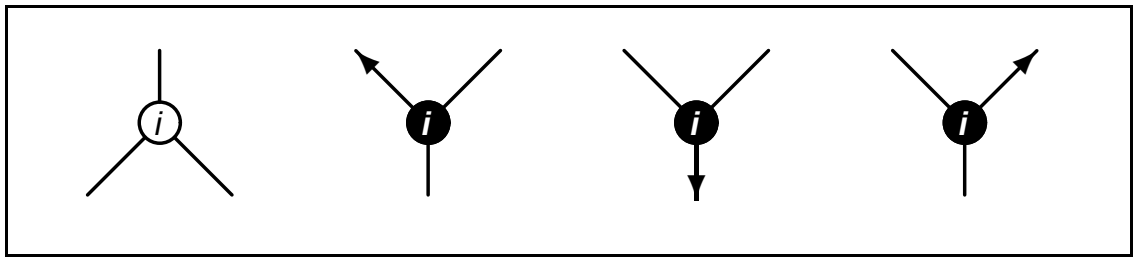
More Branches



Multimodal Categorial Grammars

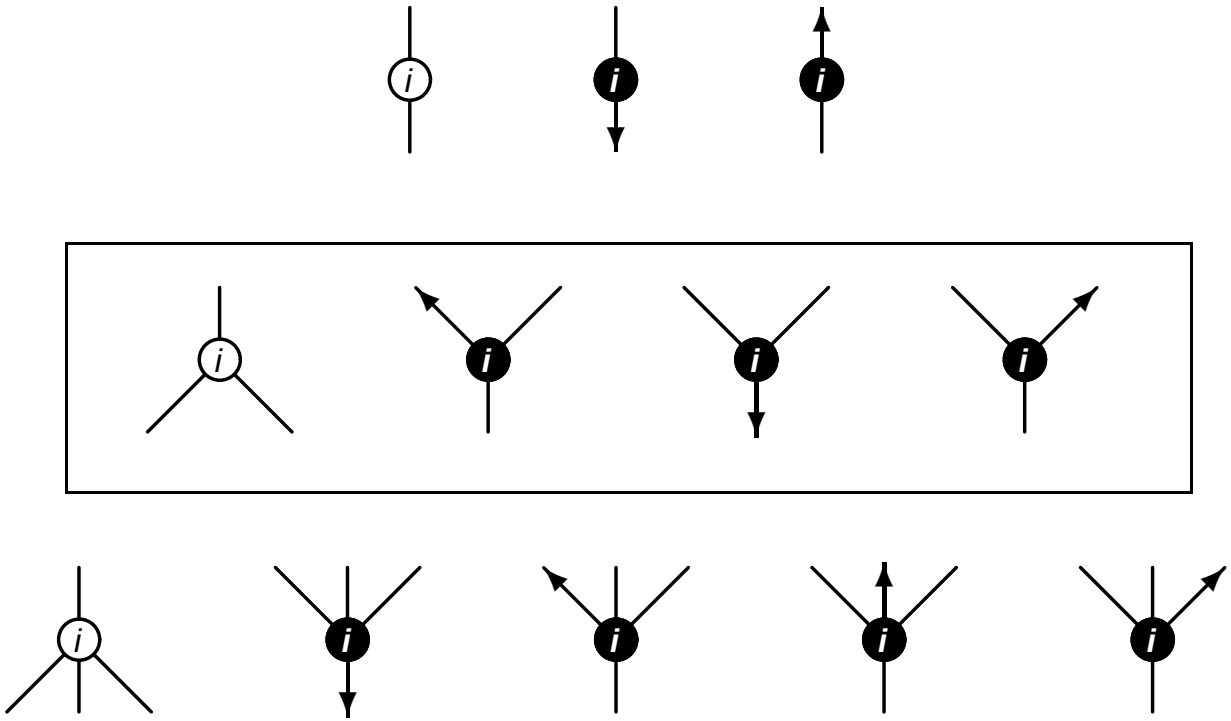
More Branches





Multimodal Categorial Grammars

More Branches

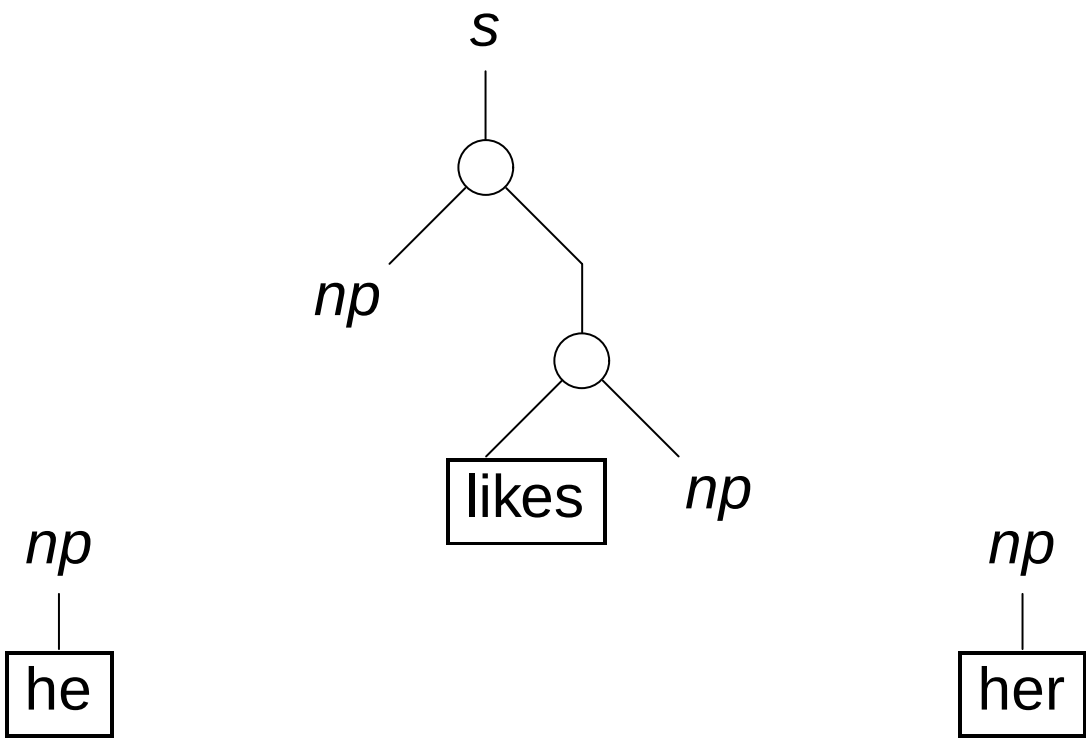


Extensions
Application: Features

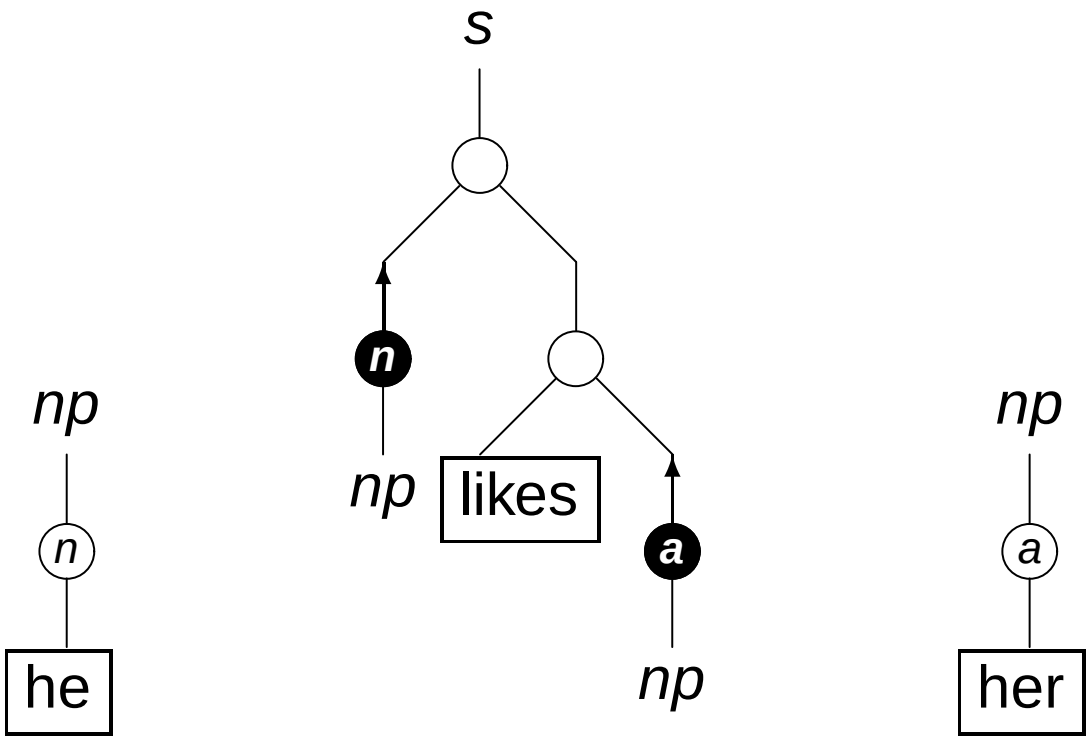
The simplest application of the unary branches is to use them to model linguistic *features*.

1. He likes her.
2. ∗He likes he.
3. ∗Her likes he.
4. ∗Her likes her.

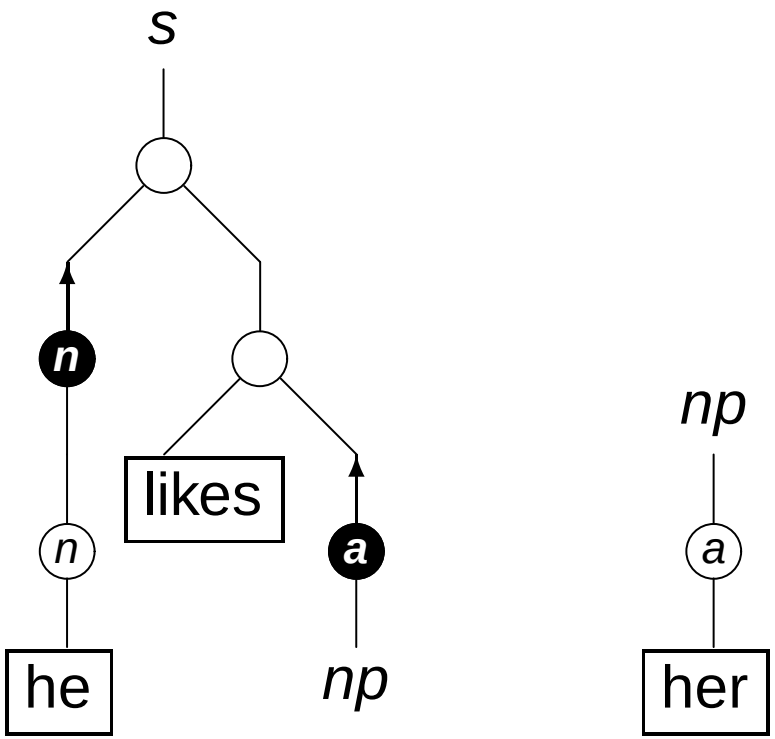
Extensions
Application: Features



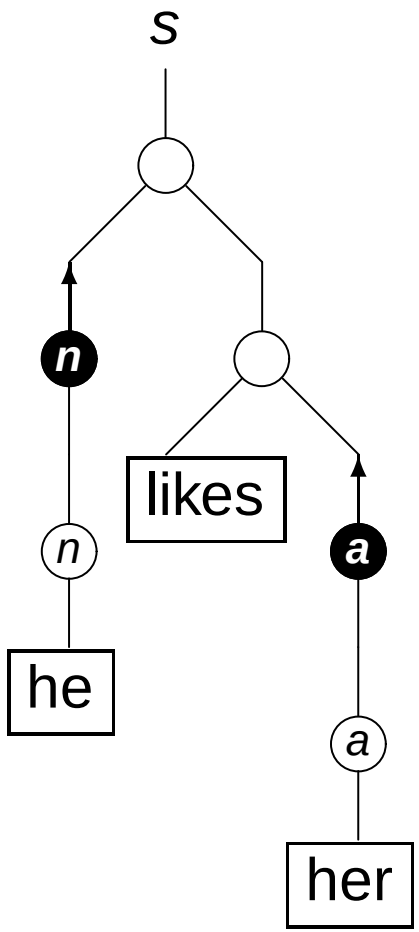
Extensions
Application: Features



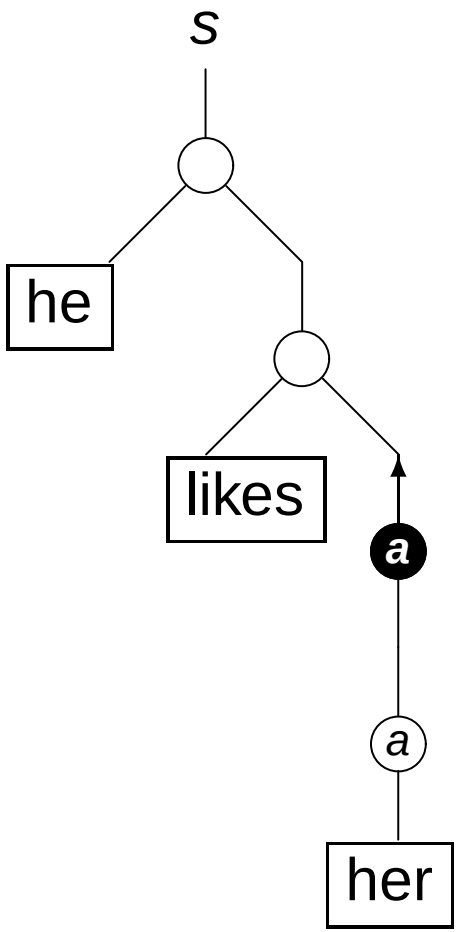
Extensions
Application: Features



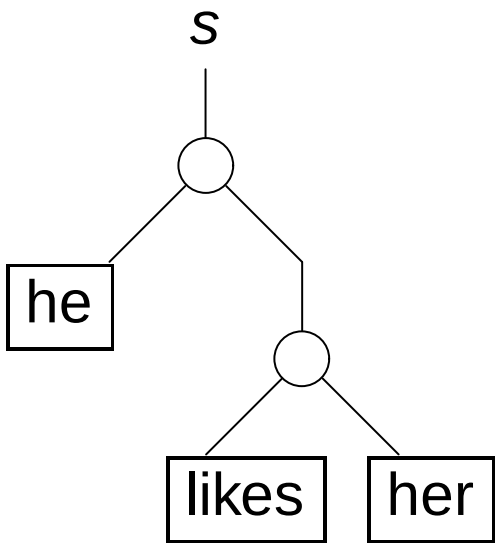
Extensions
Application: Features



Extensions
Application: Features



Extensions
Application: Features



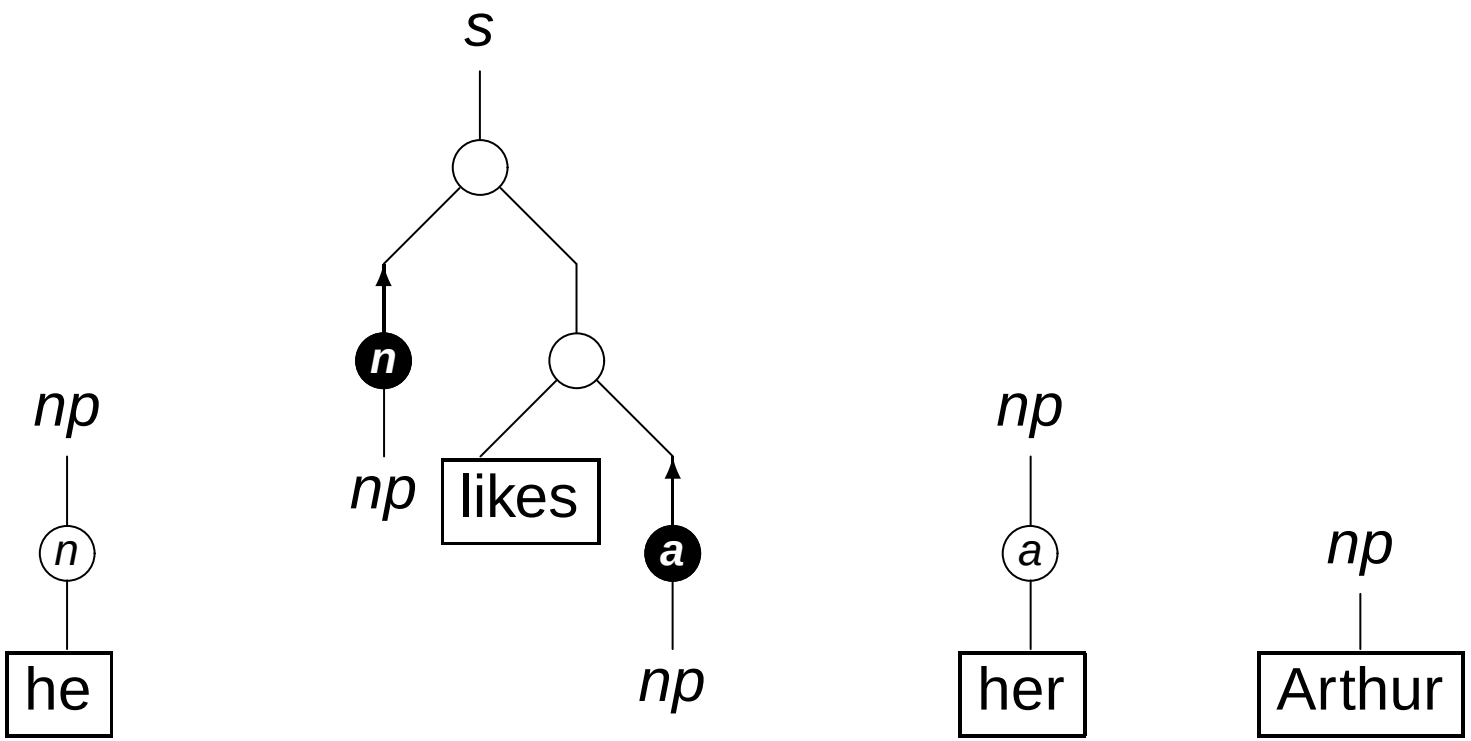
Extensions
Application: Features

We can use the unary branches to distinguish between pronouns in nominative and in accusative case.
Given this, what type do we assign to a noun like ‘Arthur’?

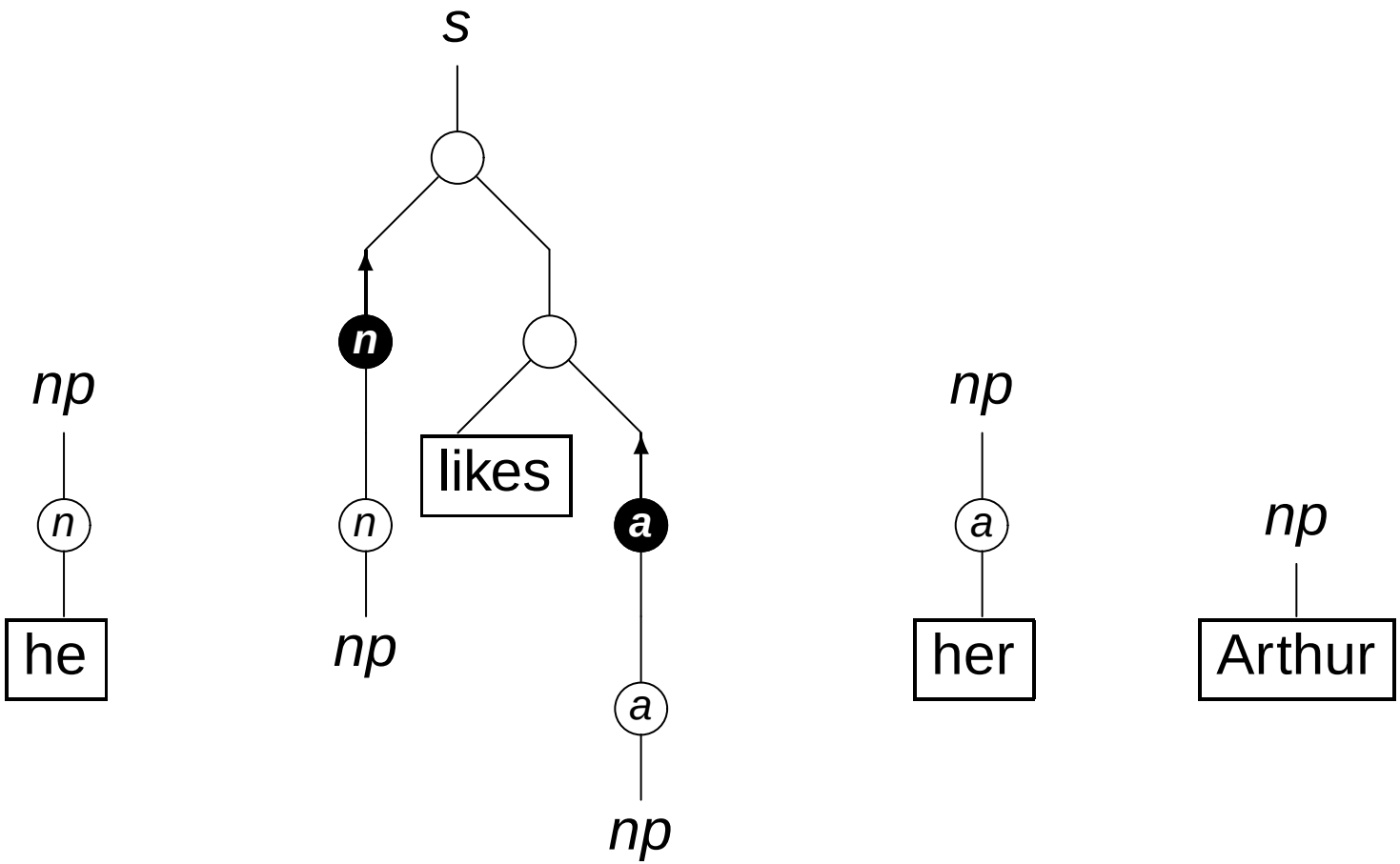
1. He likes Arthur.
2. Arthur likes her.

Assigning two types differing only in case information to every substantive in our lexicon is not very satisfying.
Is there a way to avoid this duplication?

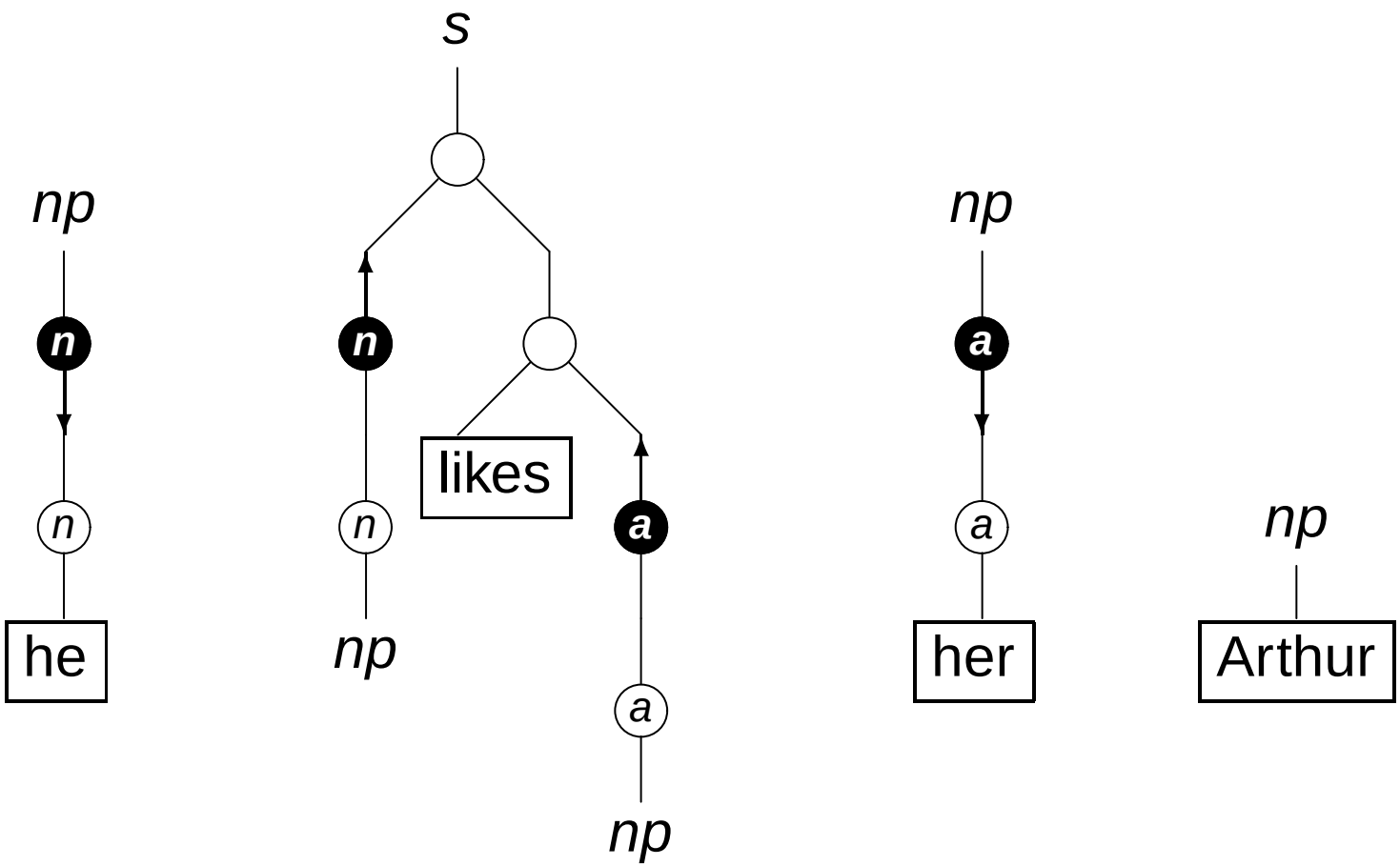
Extensions
Application: Features



Extensions
Application: Features

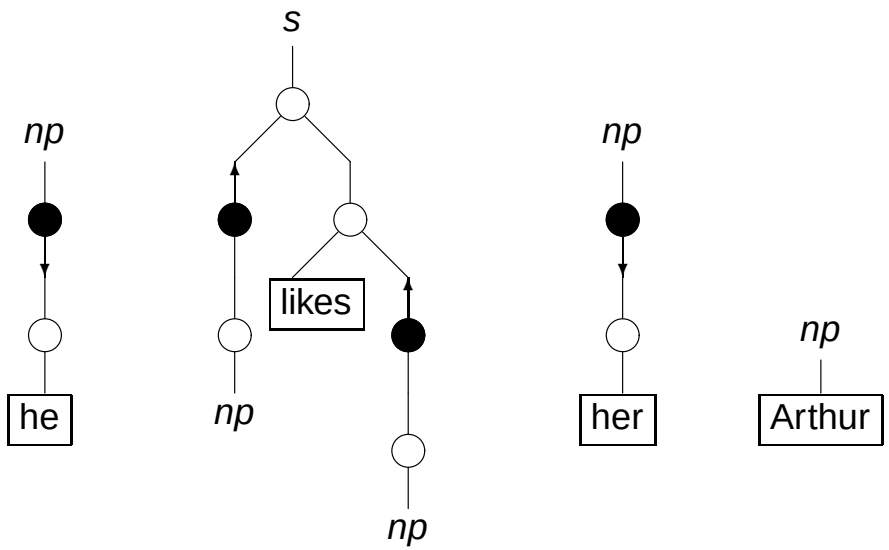


Extensions
Application: Features



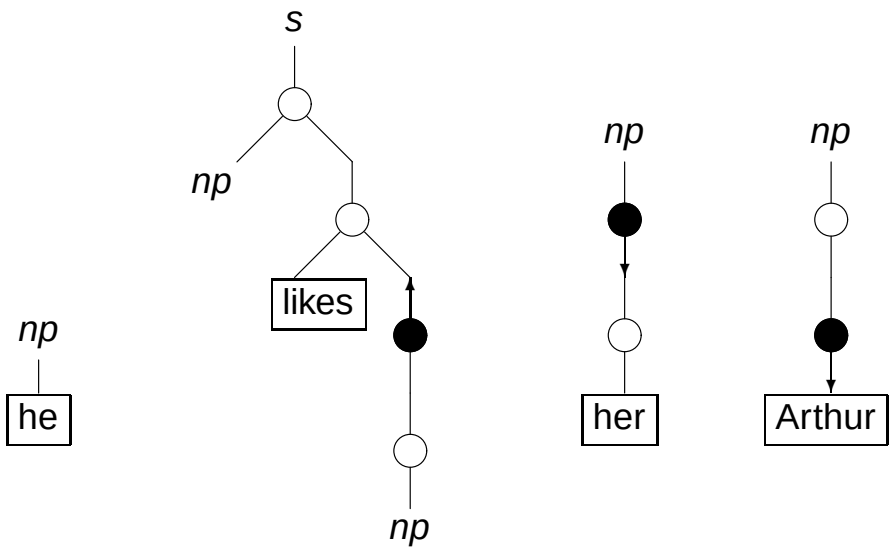
Extensions
Application: Features

Is there a solution using only one mode?



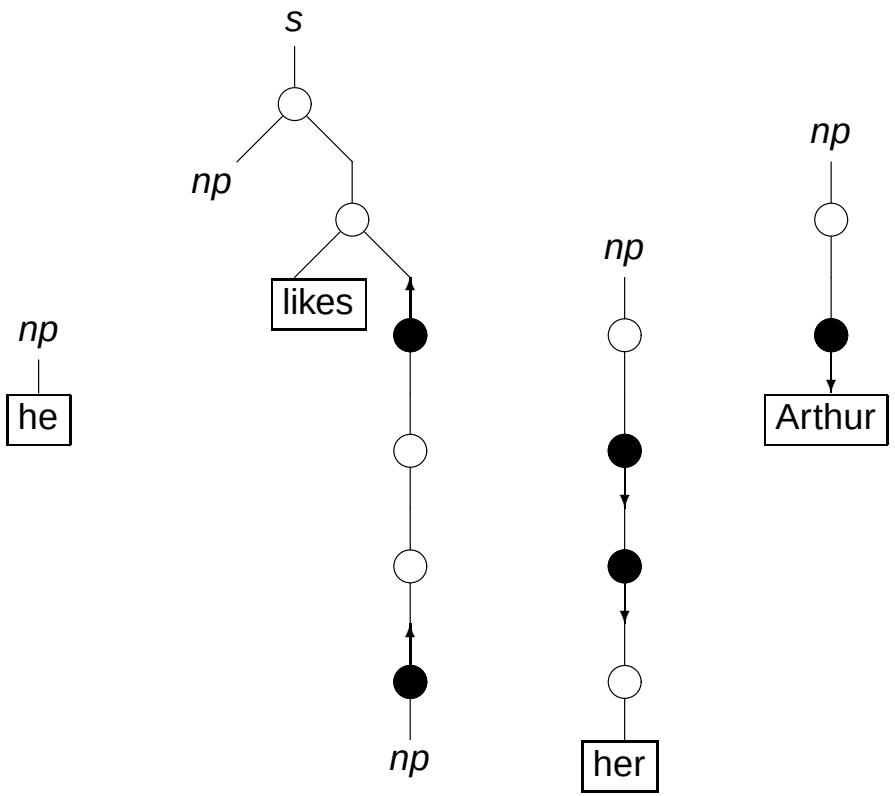
Extensions
Application: Features

Is there a solution using only one mode?



Extensions
Application: Features

Is there a solution using only one mode?



Nondeterministic Contractions

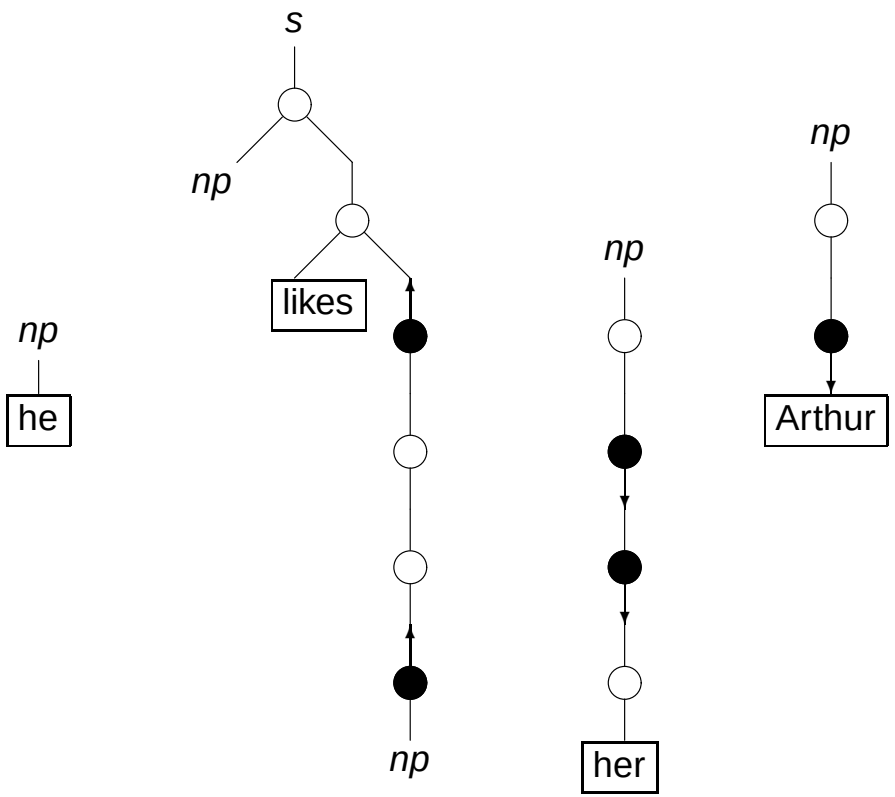
- Addition of the unary branches means the contractions can no longer be performed deterministically.
- Fortunately, we can use a small context free grammar to see if we can contract the unary modalities.

Nondeterministic Contractions

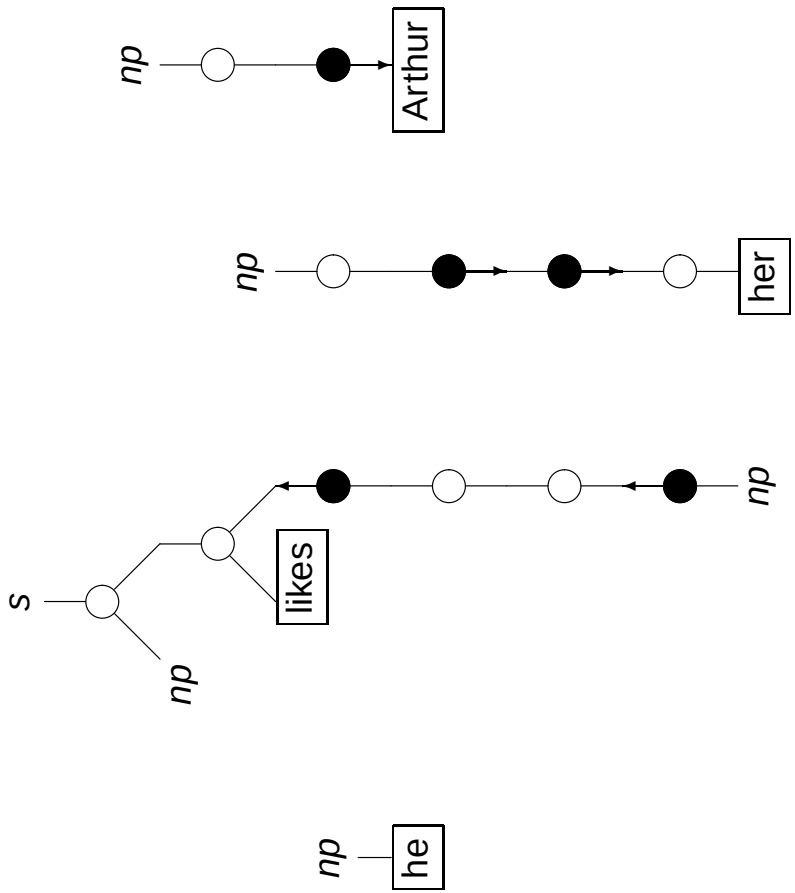
- Addition of the unary branches means the contractions can no longer be performed deterministically.
- Fortunately, we can use a small context free grammar to see if we can contract the unary modalities.

$$
\begin{aligned}
S \rightarrow\ & \epsilon \\
\mid\ & l\ S\ m\ S \\
\mid\ & m\ S\ r\ S
\end{aligned}
$$

Using the Context Free Grammar

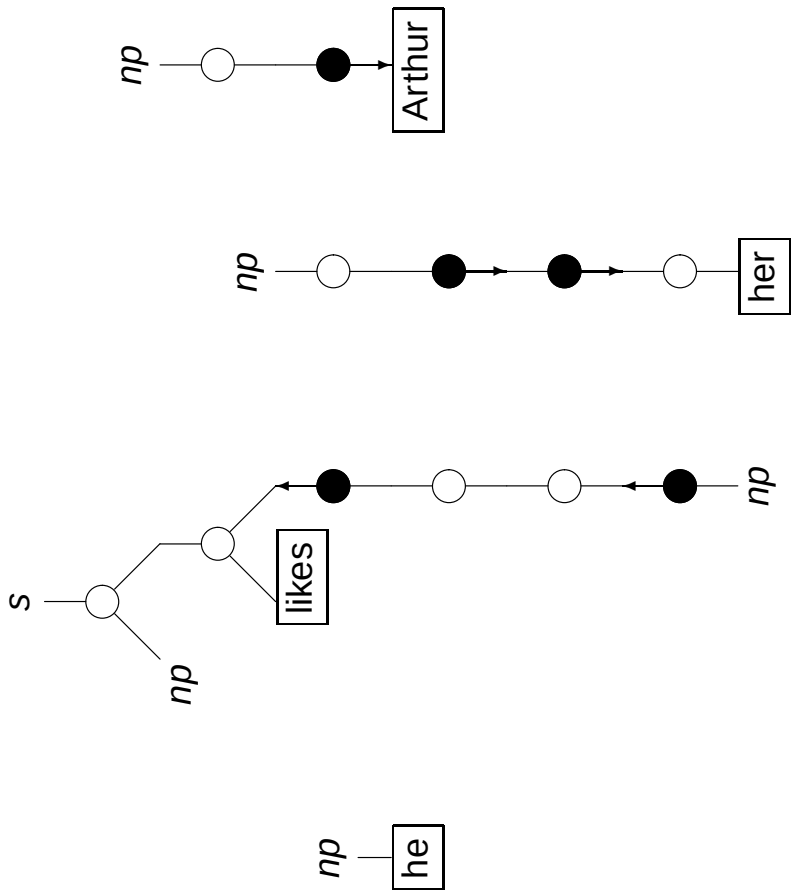


Using the Context Free Grammar



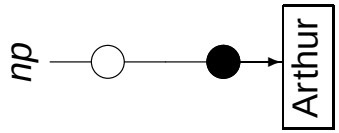
Using the Context Free Grammar

mr

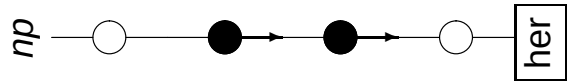


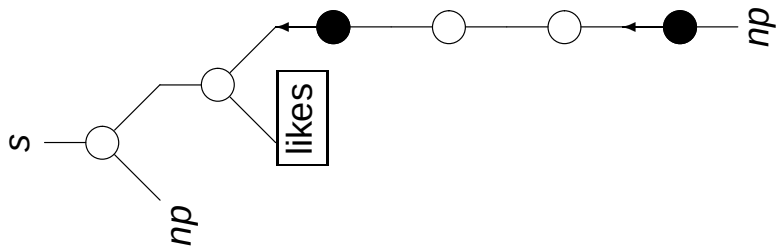
Using the Context Free Grammar

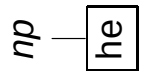
mr



mrrm

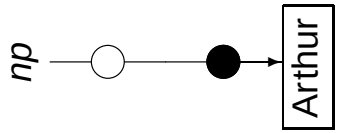




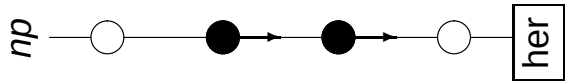


Using the Context Free Grammar

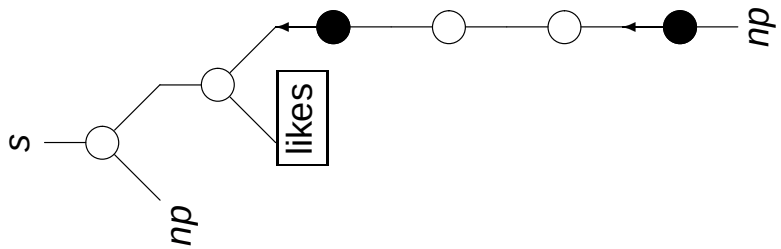
mr

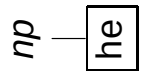


mrrm



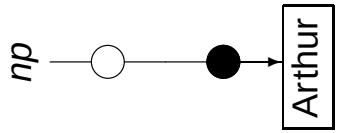
lmml



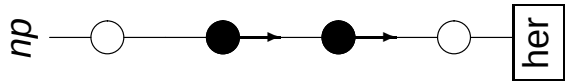


Using the Context Free Grammar

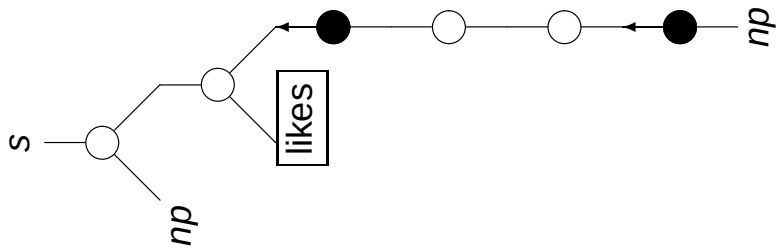
mr



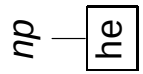
mrrm



lmml

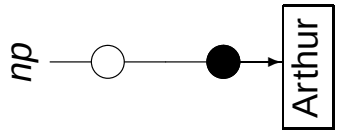


ϵ

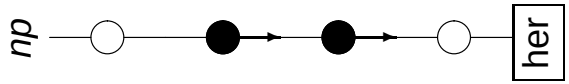


Using the Context Free Grammar

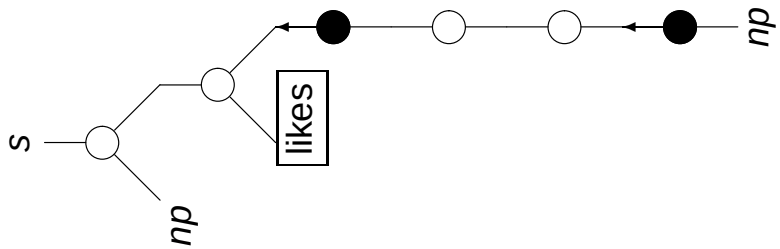
mr



mrrm

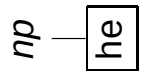


lmml



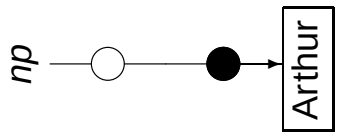
ϵ

ϵ



Using the Context Free Grammar

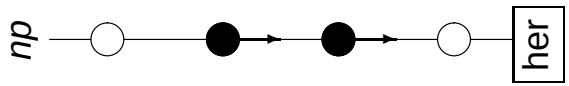
mr



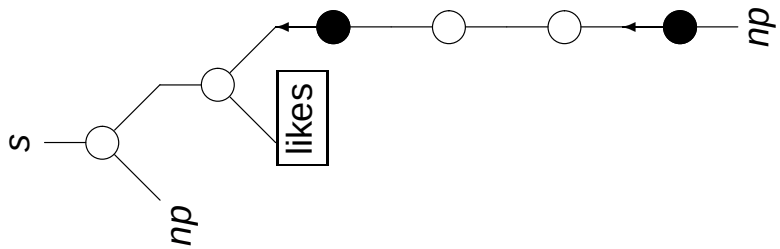
$S \rightarrow mr$

$S \rightarrow lmmlmr$

mrrm

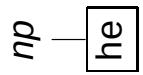


lmml



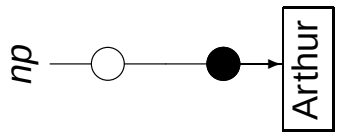
ϵ

ϵ

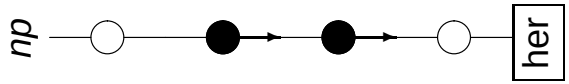


Using the Context Free Grammar

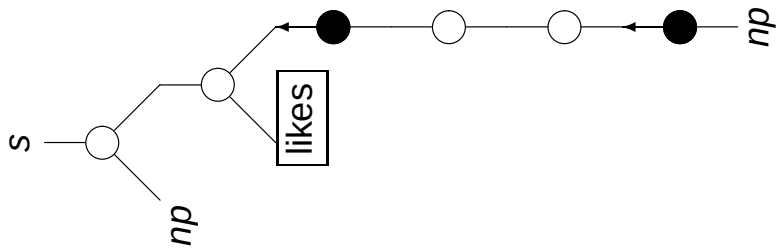
mr



mrrm

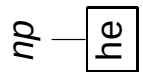


lmml



ϵ

ϵ



$S \longrightarrow mr$

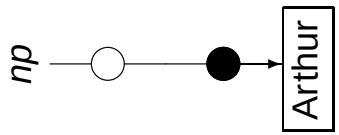
$S \longrightarrow lmmlmr$

$S \nrightarrow mrrm$

$S \longrightarrow lmmlmrrm$

Using the Context Free Grammar

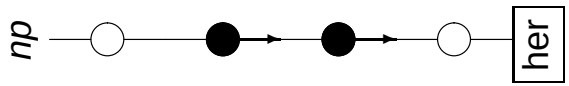
mr



$S \rightarrow mr$

$S \rightarrow lmmlmr$

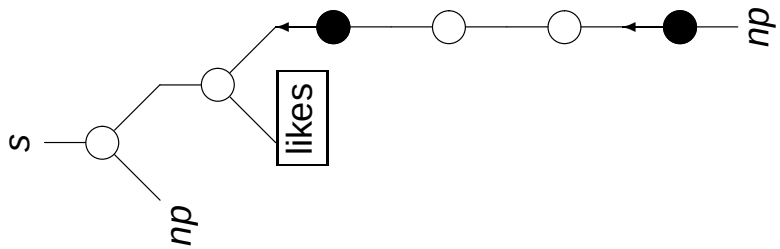
mrrm



$S \nrightarrow mrrm$

$S \rightarrow lmmlmrrm$

lmml

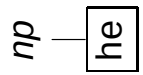


ϵ

$S \rightarrow \epsilon$

$S \nrightarrow lmml$

ϵ



Using the Context Free Grammar

$$\begin{aligned} S &\rightarrow \epsilon \\ &\mid l\ S\ m\ S \\ &\mid m\ S\ r\ S \end{aligned}$$

$$S \rightarrow l\ m\ m\ l\ m\ r$$

Using the Context Free Grammar

$$S \rightarrow \epsilon \mid l\ S\ m\ S \mid m\ S\ r\ S$$

$$S \rightarrow l\ S\ m\ S$$

$$S \rightarrow l\ m\ m\ l\ m\ r$$

Using the Context Free Grammar

$$S \rightarrow \epsilon \mid l\ S\ m\ S \mid m\ S\ r\ S$$

$$S \rightarrow l\ S\ m\ S$$
$$S \rightarrow l\ m\ S$$

$$S \rightarrow l\ m\ m\ l\ m\ r$$

Using the Context Free Grammar

$$S \rightarrow \epsilon \mid l\ S\ m\ S \mid m\ S\ r\ S$$

$$S \rightarrow l\ S\ m\ S$$
$$S \rightarrow l\ m\ S$$
$$S \rightarrow l\ m\ m\ S\ r\ S$$

$$S \rightarrow l\ m\ m\ l\ m\ r$$

Using the Context Free Grammar

$$
\begin{aligned}
S &\rightarrow \epsilon \\
&\mid l\ S\ m\ S \\
&\mid m\ S\ r\ S
\end{aligned}
$$

$$S \rightarrow l\ S\ m\ S$$
$$S \rightarrow l\ m\ S$$
$$S \rightarrow l\ m\ m\ S\ r\ S$$
$$S \rightarrow l\ m\ m\ l\ S\ m\ r\ S$$

$$S \rightarrow l\ m\ m\ l\ m\ r$$

Using the Context Free Grammar

$$
\begin{aligned}
S &\rightarrow \epsilon \\
&\mid l\ S\ m\ S \\
&\mid m\ S\ r\ S
\end{aligned}
$$

$$
\begin{aligned}
S &\rightarrow l\ S\ m\ S \\
S &\rightarrow l\ m\ S \\
S &\rightarrow l\ m\ m\ S\ r\ S \\
S &\rightarrow l\ m\ m\ l\ S\ m\ r\ S \\
S &\rightarrow l\ m\ m\ l\ m\ r\ S \\
S &\rightarrow l\ m\ m\ l\ m\ r
\end{aligned}
$$

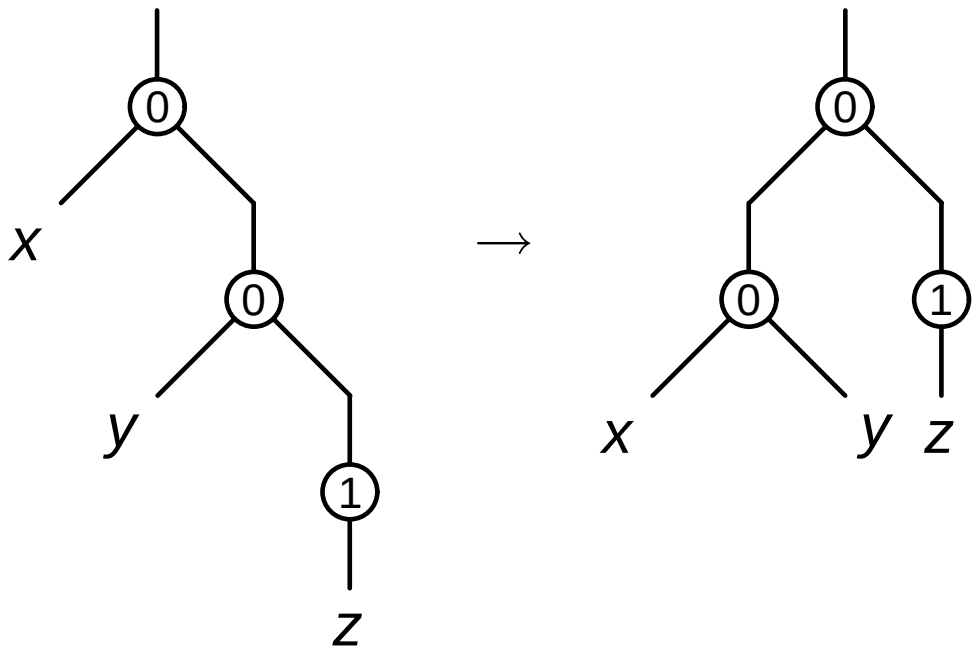
Multimodal Categorial Grammars

Structural Rules

- Even though the extensions we've discussed can be used to increase the level of detail in our grammars, they don't increase the generative capacity.
- Our final extension, the *structural rules*, which take the form of tree rewrites in the graphs, allow us to do this.

Multimodal Categorial Grammars

Structural Rules

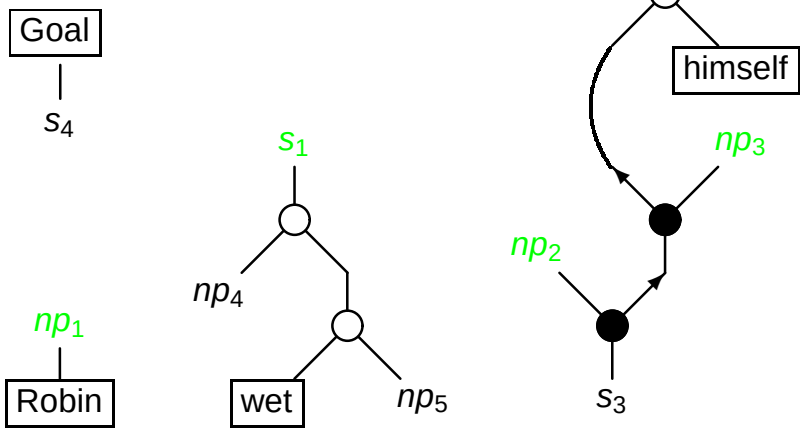


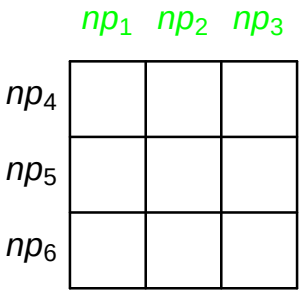
Parsing Problem

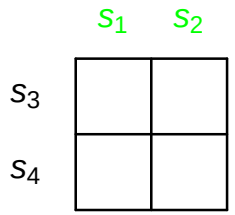
- There are $n!$ possible axiom connections to consider when parsing a sentence.
- Grail uses several strategies to reduce this number.
- The first restriction considers only connections which result in an acyclic and connected graph.
- The second restriction takes word order into account, assigning string positions to all atomic formulas.

Filtering Axiom Links

Connection Grid: Negative Atoms

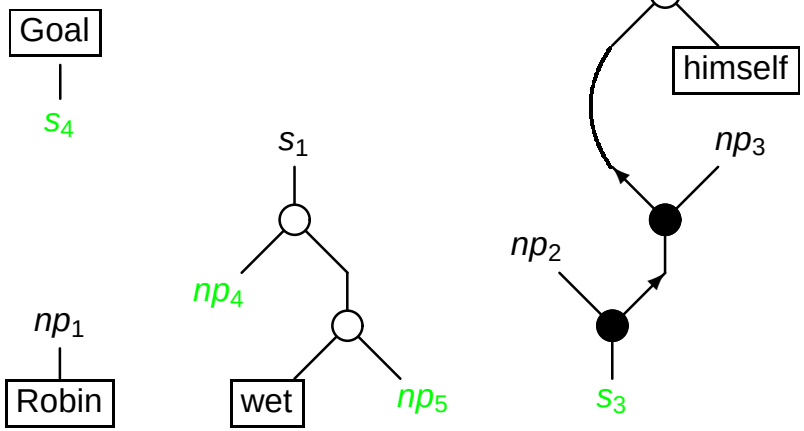


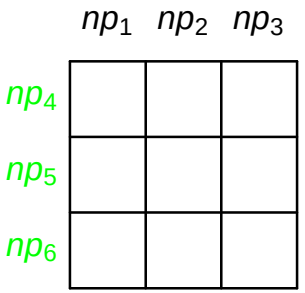


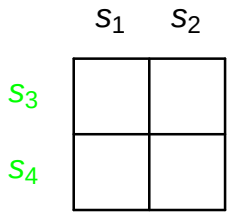


Filtering Axiom Links

Connection Grid: Positive Atoms

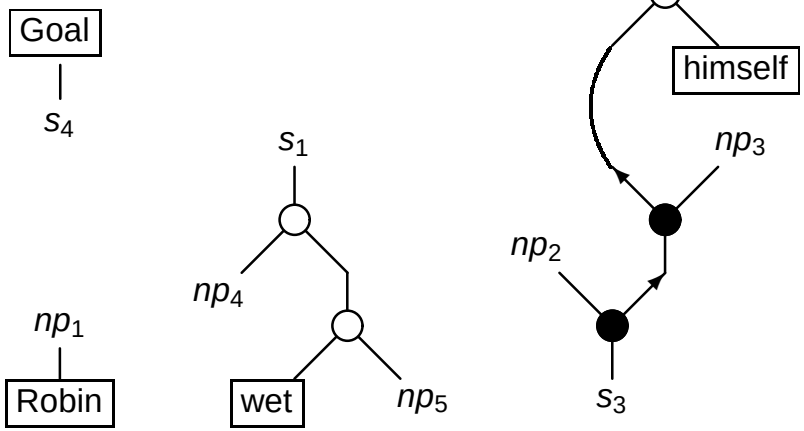


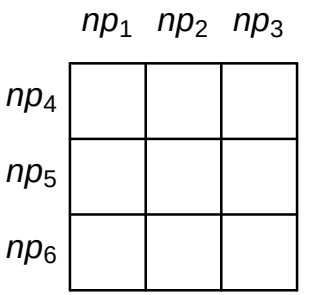


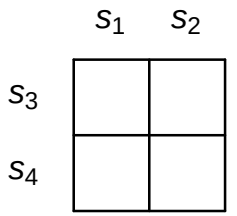


Filtering Axiom Links

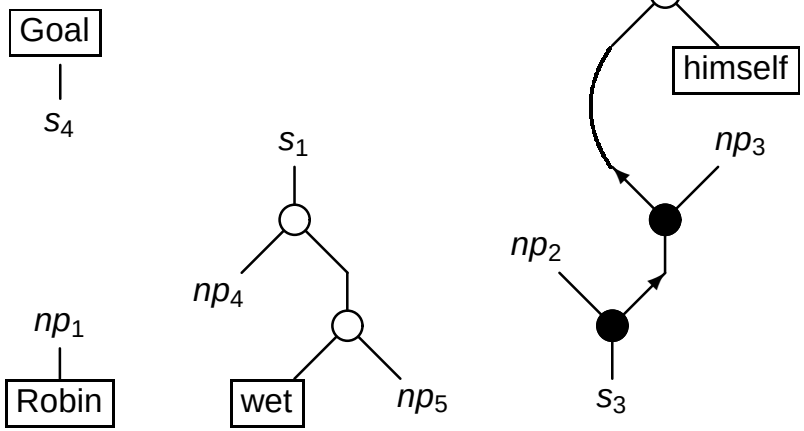
Acyclicity

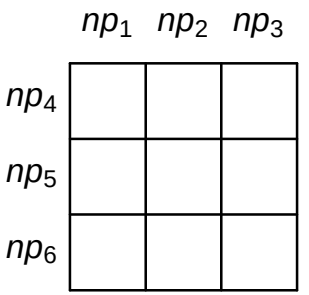


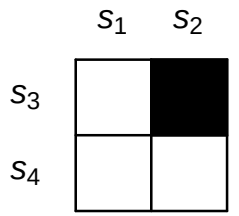




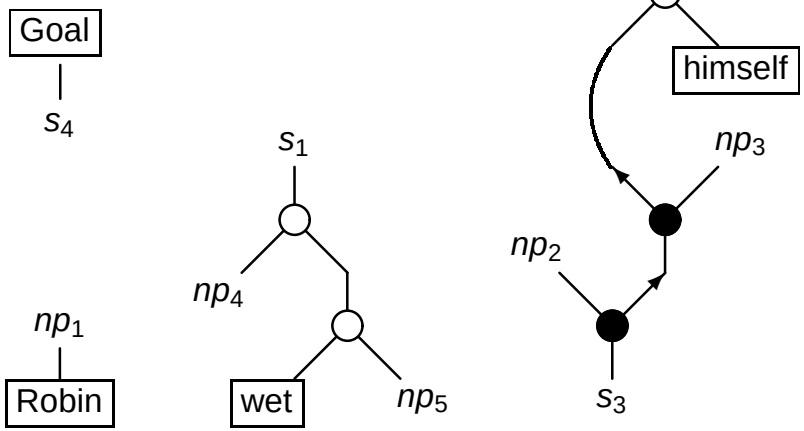
Filtering Axiom Links
Acyclicity

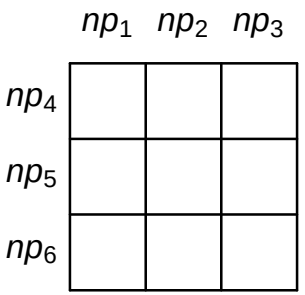


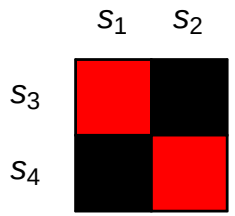




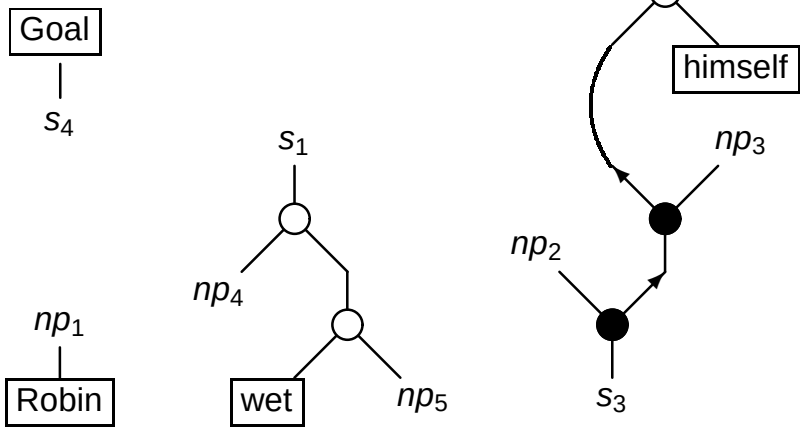
Filtering Axiom Links
Acyclicity

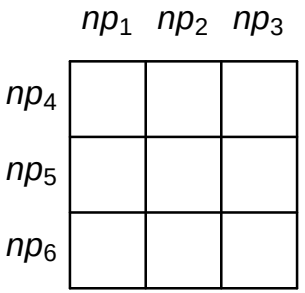


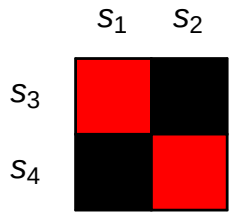




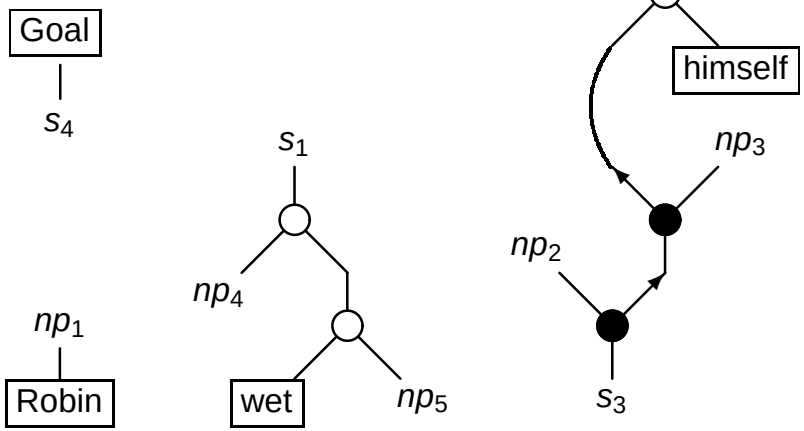
Filtering Axiom Links
Connectedness

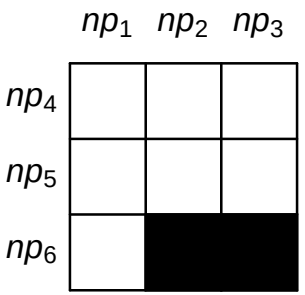


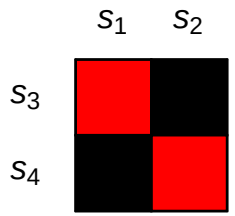




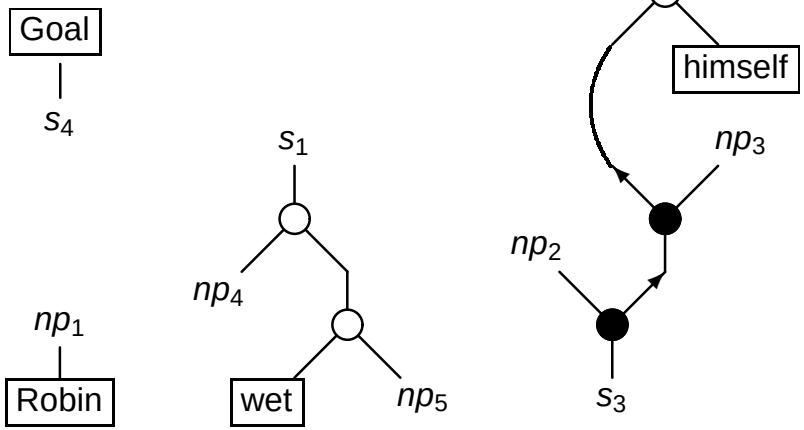
Filtering Axiom Links
Connectedness

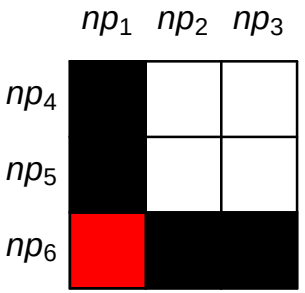


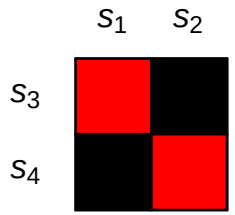




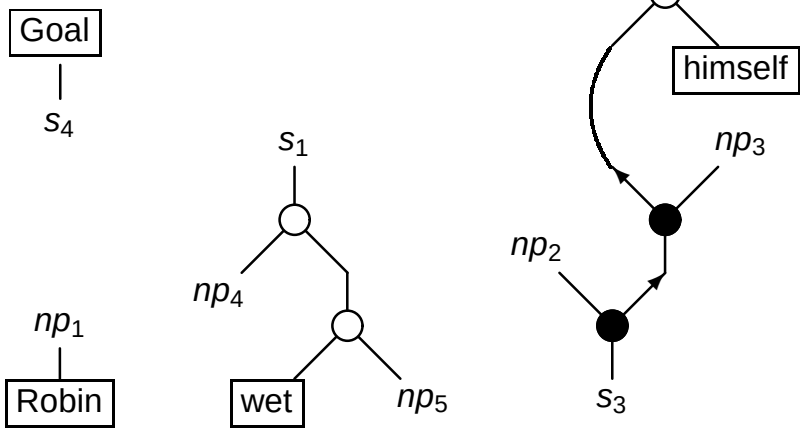
Filtering Axiom Links
Connectedness

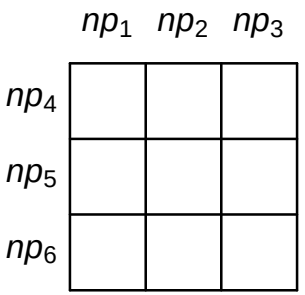


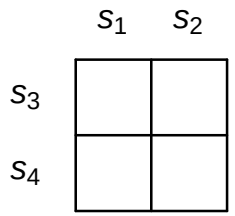




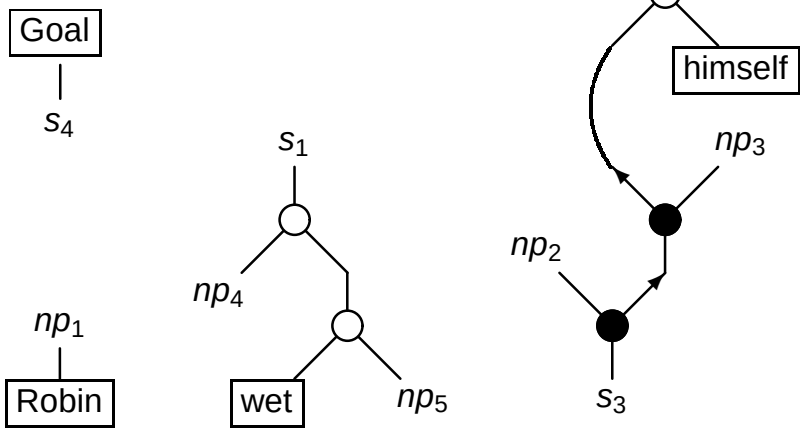
Filtering Axiom Links
Word Order

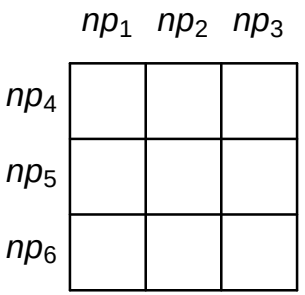


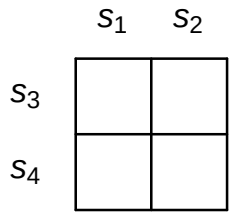


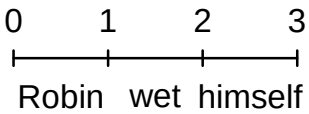


Filtering Axiom Links
Word Order

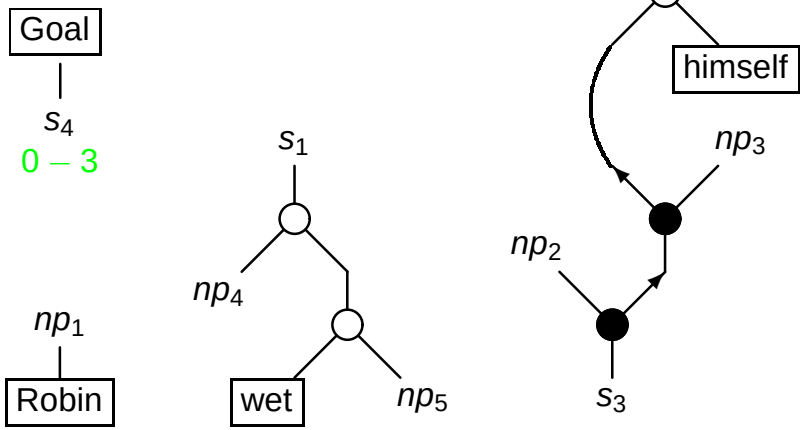


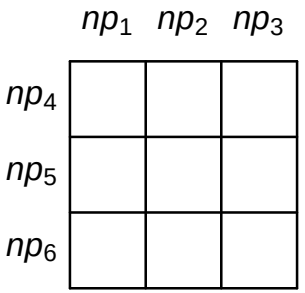


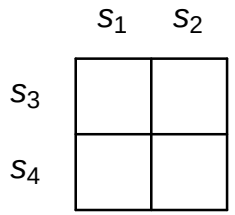


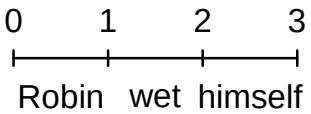


Filtering Axiom Links
Word Order

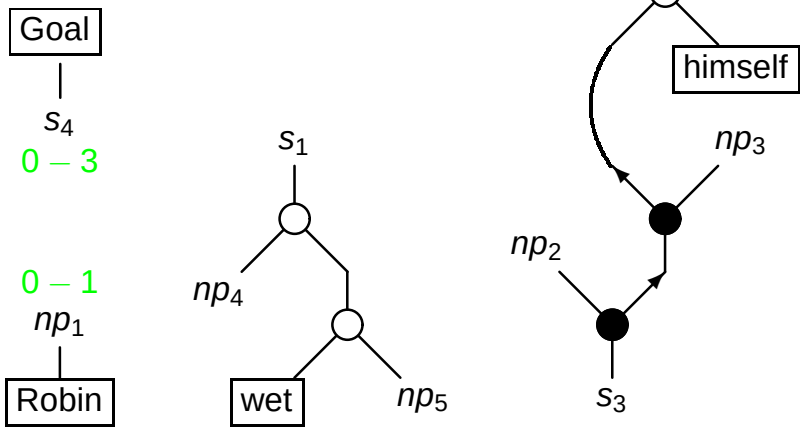


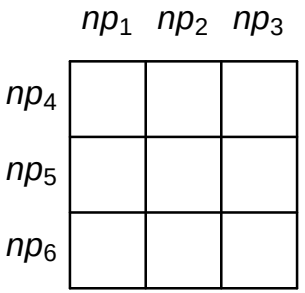


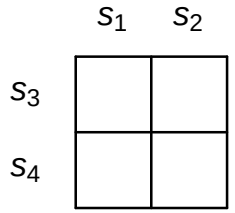


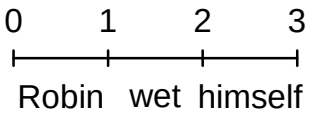


Filtering Axiom Links
Word Order

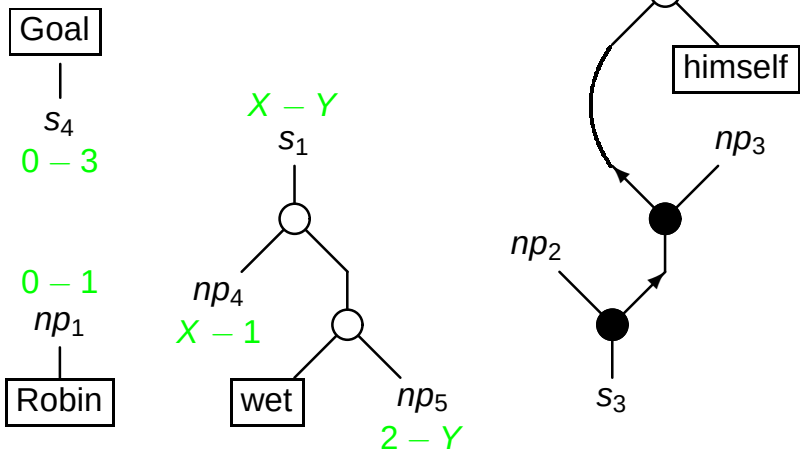


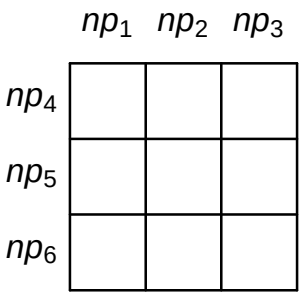


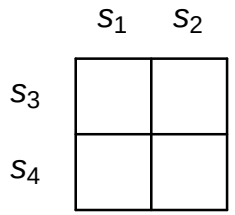


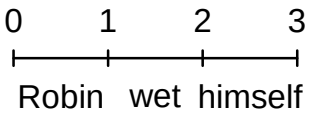


Filtering Axiom Links
Word Order

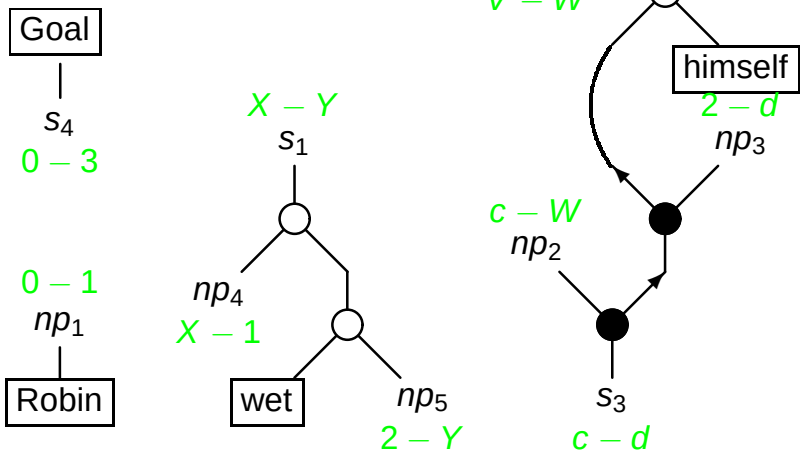


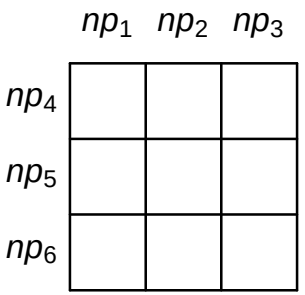


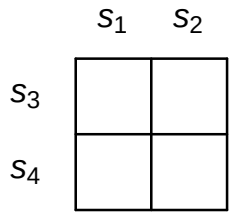


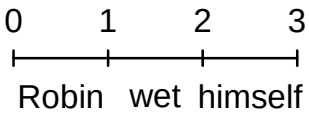


Filtering Axiom Links
Word Order

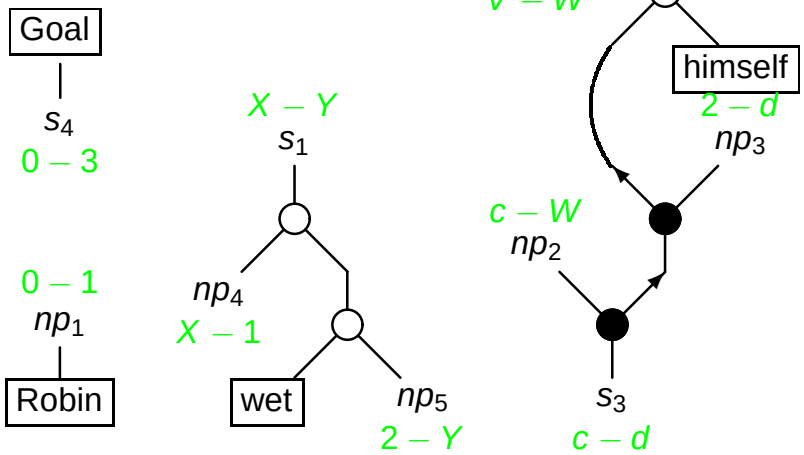


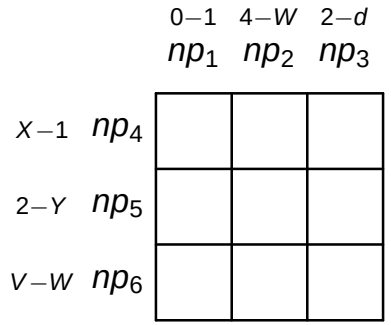


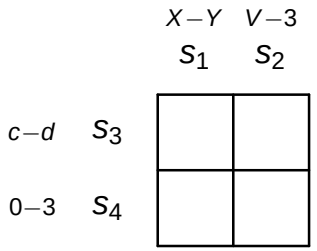


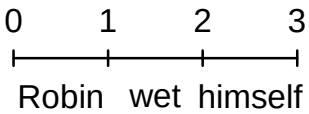


Filtering Axiom Links
Word Order

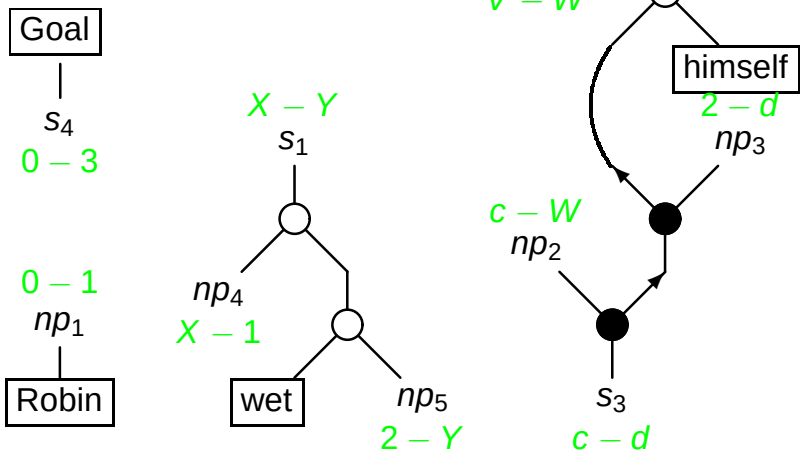


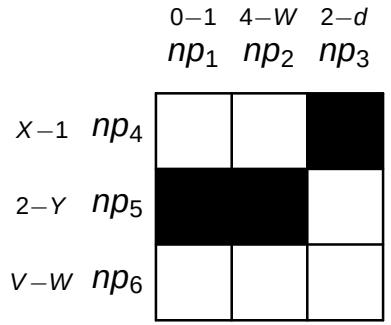


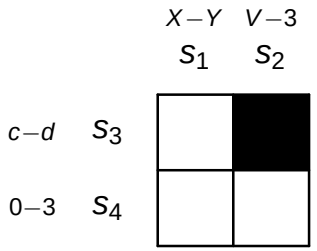


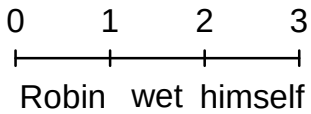


Filtering Axiom Links
Word Order



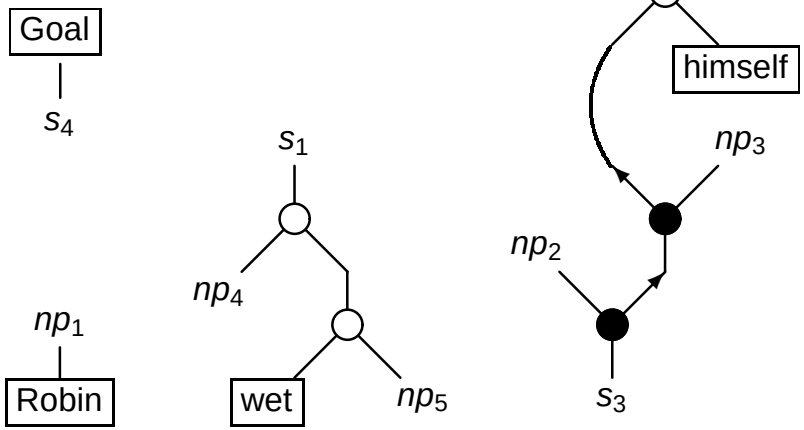


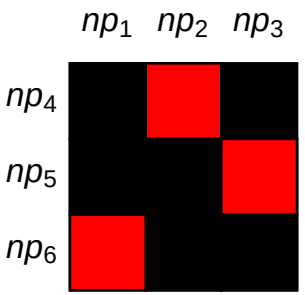


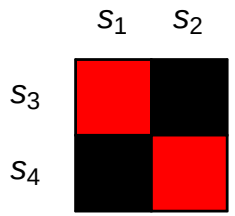


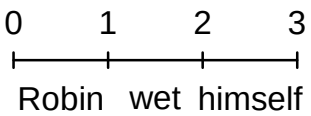
Filtering Axiom Links

Word Order, Acyclicity and Connectedness Together



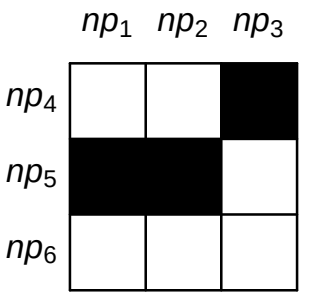


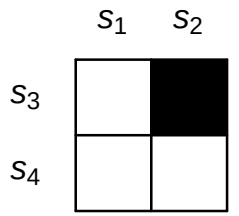




Filtering Axiom Links

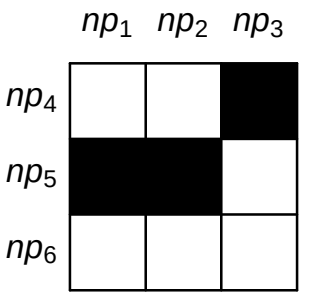
A Step Back: What's Wrong With This Picture?

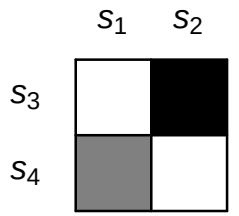




Filtering Axiom Links

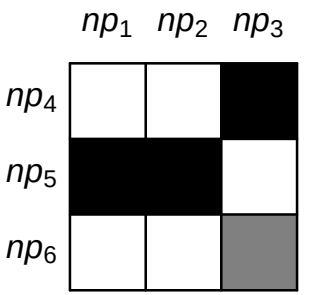
A Step Back: What's Wrong With This Picture?

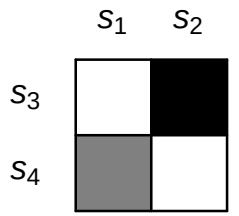




Filtering Axiom Links

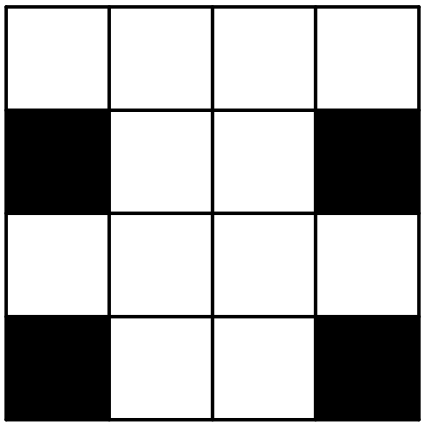
A Step Back: What's Wrong With This Picture?





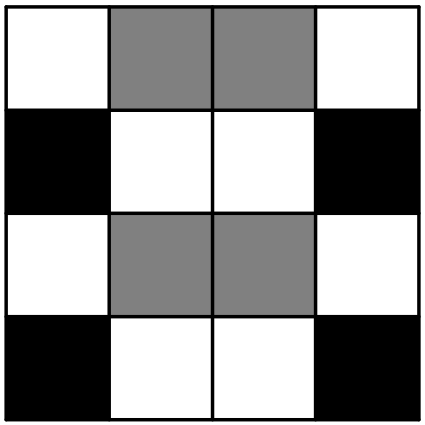
Filtering Axiom Links

What's Wrong With This Picture?



Filtering Axiom Links

What's Wrong With This Picture?



Filtering Axiom Links
Coherence

- Some links do not belong to any perfect matching, so we can remove them.
- Some links belong to *every* perfect matching, so we can remove all alternatives.
- How do we decide which is which?

Filtering Axiom Links

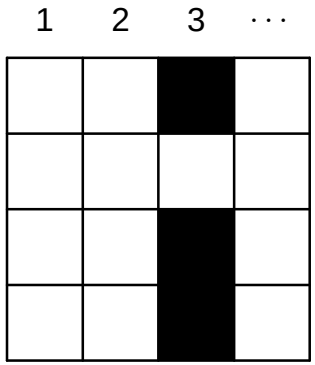
Coherence: An Easier Problem

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

Filtering Axiom Links

Coherence: An Easier Problem

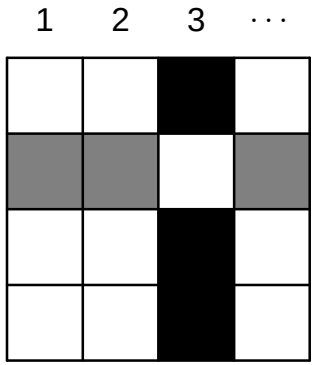
9				7	4			
		6	5				3	
	5							7
	3	8						9
		1	6		8	7		
4						3	5	
6							2	
	1				7	8		
			4	9				3



Filtering Axiom Links

Coherence: An Easier Problem

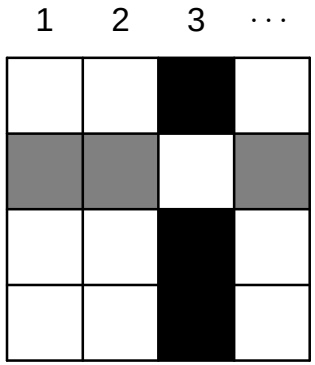
9				7	4			
		6	5				3	
	5							7
	3	8						9
		1	6	3	8	7		
4						3	5	
6							2	
	1				7	8		
			4	9				3



Filtering Axiom Links

Coherence: An Easier Problem

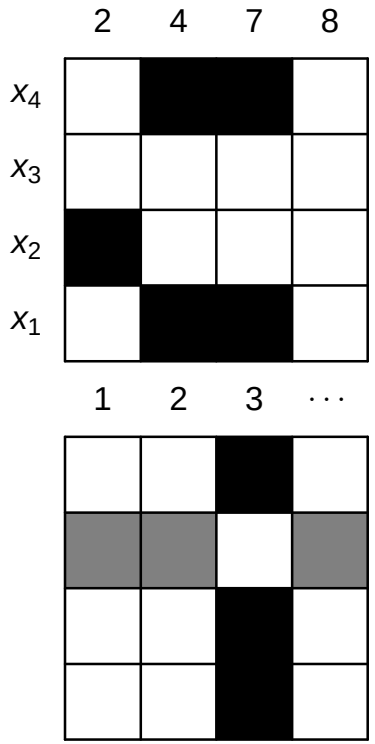
9				7	4			
		6	5				3	
	5							7
	3	8		4	5			9
5	9	1	6	3	8	7	4	2
4	6					3	5	8
6							2	
	1				7	8		
			4	9			7	3



Filtering Axiom Links

Coherence: An Easier Problem

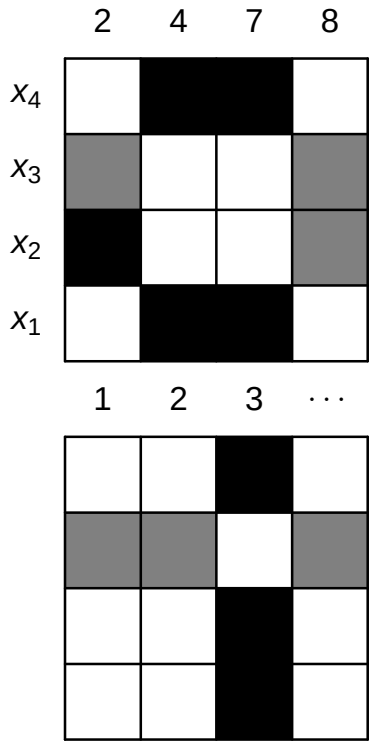
9	x_4			7	4			
	x_3	6	5				3	
	5							7
	3	8		4	5			9
5	9	1	6	3	8	7	4	2
4	6					3	5	8
6	x_2						2	
	1				7	8		
	x_1		4	9			7	3



Filtering Axiom Links

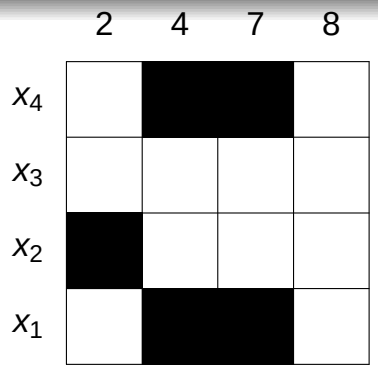
Coherence: An Easier Problem

9	x_4			7	4			
	x_3	6	5				3	
	5							7
	3	8		4	5			9
5	9	1	6	3	8	7	4	2
4	6					3	5	8
6	x_2						2	
	1				7	8		
	x_1		4	9			7	3



Filtering Axiom Links

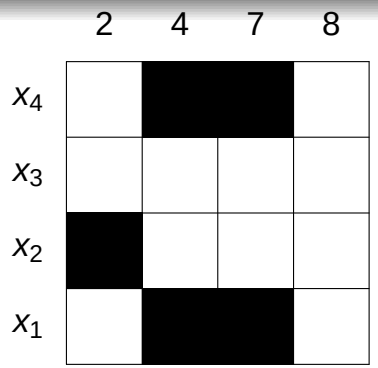
Coherence: Regin's Algorithm



Regin proposes the following algorithm for eliminating edges from a matching problem.

Filtering Axiom Links

Coherence: Regin's Algorithm

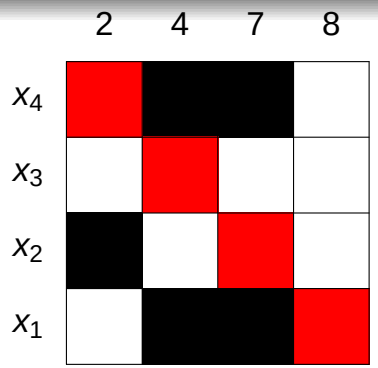


Regin proposes the following algorithm for eliminating edges from a matching problem.

1. compute a perfect matching; if none exists, fail.

Filtering Axiom Links

Coherence: Regin's Algorithm

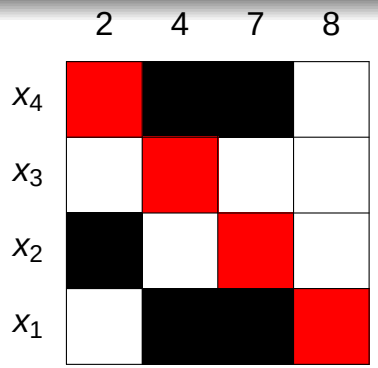


Regin proposes the following algorithm for eliminating edges from a matching problem.

1. compute a perfect matching; if none exists, fail.

Filtering Axiom Links

Coherence: Regin's Algorithm

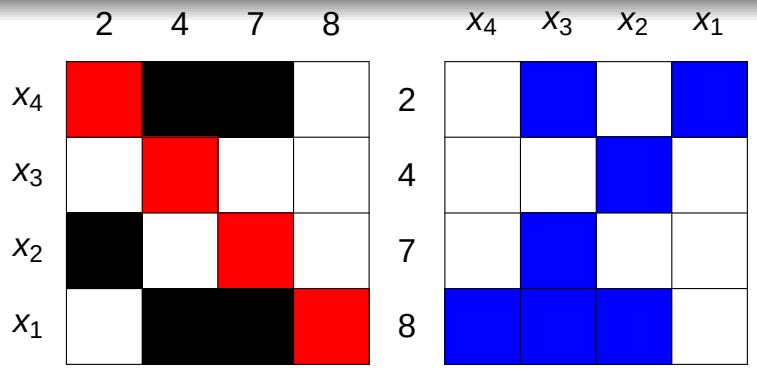


Regin proposes the following algorithm for eliminating edges from a matching problem.

1. compute a perfect matching; if none exists, fail.
2. using links which are alternatively inside and the reverse of links outside the matching, compute the strong components.

Filtering Axiom Links

Coherence: Regin's Algorithm

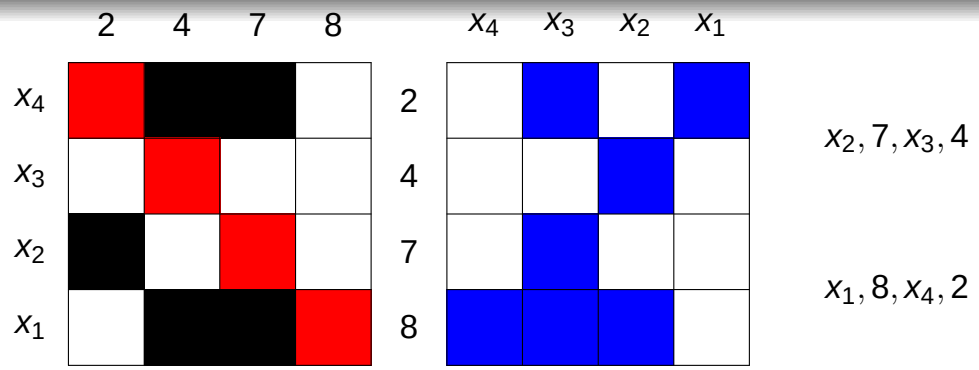


Regin proposes the following algorithm for eliminating edges from a matching problem.

1. compute a perfect matching; if none exists, fail.
2. using links which are alternatively inside and the reverse of links outside the matching, compute the strong components.

Filtering Axiom Links

Coherence: Regin's Algorithm

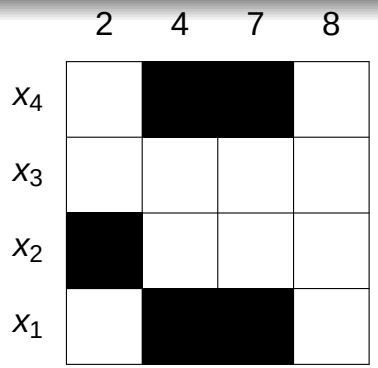


Regin proposes the following algorithm for eliminating edges from a matching problem.

1. compute a perfect matching; if none exists, fail.
2. using links which are alternatively inside and the reverse of links outside the matching, compute the strong components.

Filtering Axiom Links

Coherence: Regin's Algorithm



$x_2, 7, x_3, 4$

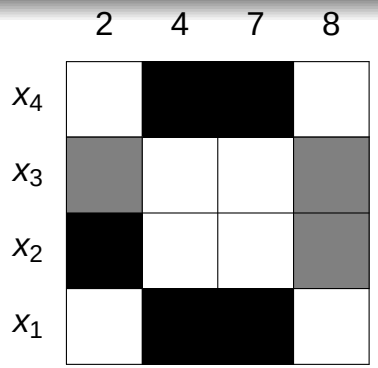
$x_1, 8, x_4, 2$

Regin proposes the following algorithm for eliminating edges from a matching problem.

1. compute a perfect matching; if none exists, fail.
2. using links which are alternatively inside and the reverse of links outside the matching, compute the strong components.
3. delete all edges which connect two different strong components.

Filtering Axiom Links

Coherence: Regin's Algorithm



$x_2, 7, x_3, 4$

$x_1, 8, x_4, 2$

Regin proposes the following algorithm for eliminating edges from a matching problem.

1. compute a perfect matching; if none exists, fail.
2. using links which are alternatively inside and the reverse of links outside the matching, compute the strong components.
3. delete all edges which connect two different strong components.

Filtering Axiom Links
Unary Modalities

- The unary modalities allow us to exploit the derivability relations between types in different ways.
- Applications include case, quantifier scope restrictions, negative polarity items and clitics.
- We can use a simple CFG to decide the derivability relation, or — for a given grammar — we can just precompile it.

Filtering Axiom Links
Unary Modalities

$$\Box\Diamond\Diamond\Box A$$

$$\diamond\diamond\Box\Box A \vdash \diamond\Box A \vdash A \vdash \Box\Diamond A \vdash \Box\Box\Diamond\Diamond A$$

$$\Diamond\Box\Box\Diamond A$$

- The unary modalities allow us to exploit the derivability relations between types in different ways.
- Applications include case, quantifier scope restrictions, negative polarity items and clitics.
- We can use a simple CFG to decide the derivability relation, or — for a given grammar — we can just precompile it.

Filtering Axiom Links
Complexity

- $O(n^4)$ acyclicity and connectedness

Filtering Axiom Links
Complexity

- $O(n^4)$ acyclicity and connectedness
- $O(n^2)$ word order

Filtering Axiom Links
Complexity

- $O(n^4)$ acyclicity and connectedness
- $O(n^2)$ word order
- $O(n^2)$ unary modalities (if precompiled)

Filtering Axiom Links
Complexity

- $O(n^4)$ acyclicity and connectedness
- $O(n^2)$ word order
- $O(n^2)$ unary modalities (if precompiled)
- $O(n^4)$ Regin's algorithm

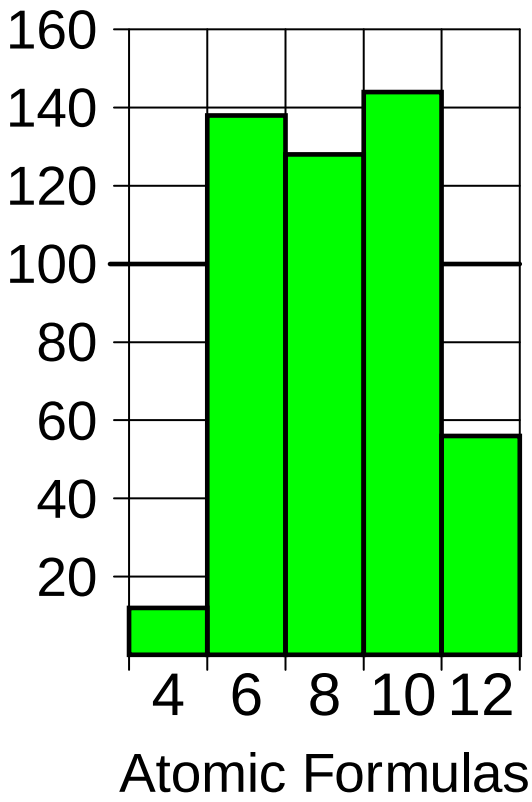
Filtering Axiom Links
Complexity

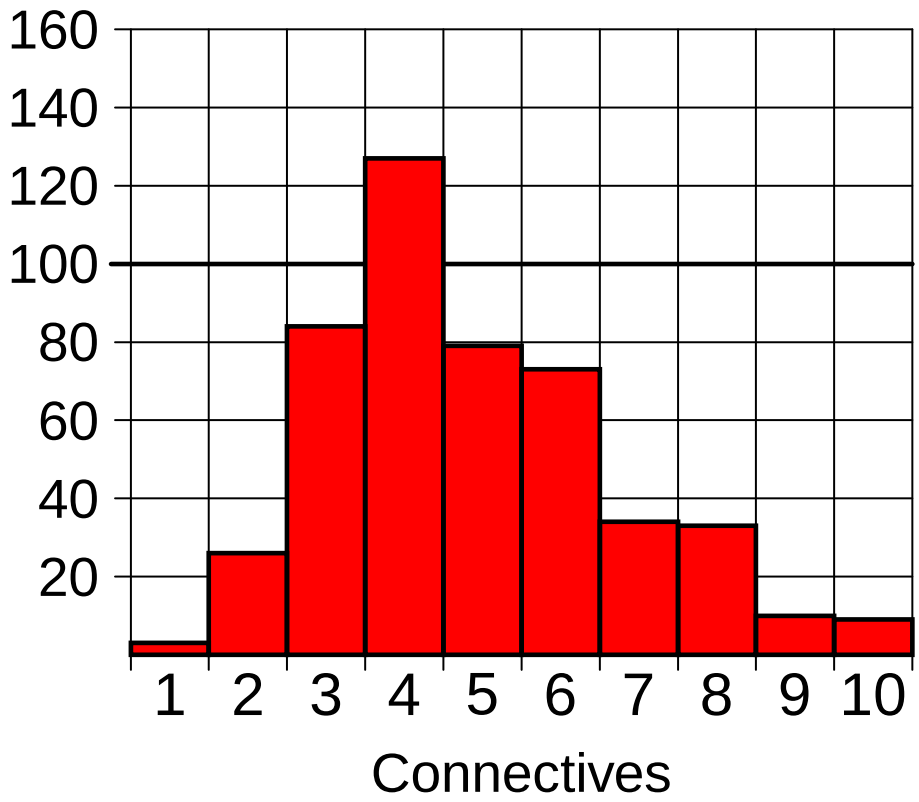
- $O(n^4)$ acyclicity and connectedness
- $O(n^2)$ word order
- $O(n^2)$ unary modalities (if precompiled)
- $O(n^4)$ Regin's algorithm
- $O(n^4)$ Total complexity

Evaluation
Random L Sequents

- In order to evaluate the different filtering strategies, I have randomly generated a set of 478 derivable L sequents, containing a total of 4.012 atomic formulas and a total of 2.330 connectives.
- This makes a total of 15.946 possible planar axiom links.
- Of these, 2.546 correspond to different proofs.

Evaluation
Random L Sequents





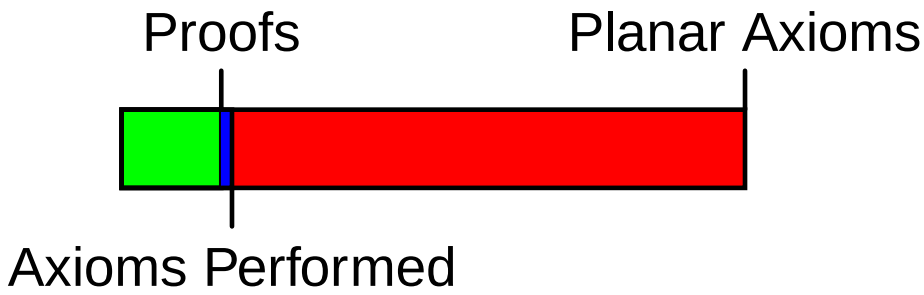
Evaluation
Random L Sequents

- 15.946 total possible planar axioms

Evaluation
Random L Sequents

- 15.946 total possible planar axioms
- 2.546 axioms belong to some proof
- 13.400 do not belong to any proof

Evaluation
Random L Sequents



- 15.946 total possible planar axioms
- 2.546 axioms belong to some proof
- 13.400 do not belong to any proof
- all but 279 of these are excluded (97.92% of total erroneous links excluded)

Conclusions and Future Work

- Multimodal extensions of NL increase the generative capacity without sacrificing the logical nature of the system.
- Using different constraints allows us to consider just a reasonable number of axiom links.
- Next step: take on some large corpus-induced grammars.