Modelling Syntactic Gradience with Loose Constraint-based Parsing

Modélisation de la gradience syntaxique par analyse relâchée à base de contraintes

Jean-Philippe Prost

LPL, Université de Provence, Aix-en-Provence
CLT, Macquarie University, Sydney

10 December 2008

Motivation

Graded Grammaticality

- Je soutiens ma thèse de doctorat
- *Je soutiens mon thèse de le doctorat
- *Moi thèse soutenir de le doctorat mon

- *I am the chair of my department*
- **Me are the chair of my department*
- **Me are chair the me's department of*

(Pullum and Scholz, 2001)

Motivation

An adequate linguistic theory will have to recognize degrees of grammaticalness

(Chomsky, 1975)

Anyone who knows a natural language knows that some utterances are not completely well formed. (...) experienced users of a language are also aware that some ungrammatical utterances are much closer to being grammatical than others.

(Pullum and Scholz, 2001)

Motivation

- How to represent and analyse graded syntactic phenomena?

Motivation

- How to represent and analyse graded syntactic phenomena?
- How to automate the computation of the degree of acceptability of an utterance?

Outline

Outline

Gradience

- Graded phenomena,

Gradience

- Graded phenomena,
- in classification problems

Example

- Is a three-legged horse a horse?
- Is a *penguin* less of a bird than a *robin*?

Modelling Gradience

Bas Aarts, 2007

- Intersective Gradience (IG)

- Subsective Gradience (SG)

Modelling Gradience

Bas Aarts, 2007

- Intersective Gradience (IG)

Example

- Is a *tomato* a *fruit* or a *vegetable*?
- Is a *van* a *car* or a *truck*?

- Subsective Gradience (SG)

Modelling Gradience

Bas Aarts, 2007

- Intersective Gradience (IG)

Example

- Is a *tomato* a *fruit* or a *vegetable*?
- Is a *van* a *car* or a *truck*?

- Subsective Gradience (SG)

Example

- Is a *penguin* less of a bird than a *robin*?
- Is a three-legged horse a horse?

Intersective Gradience

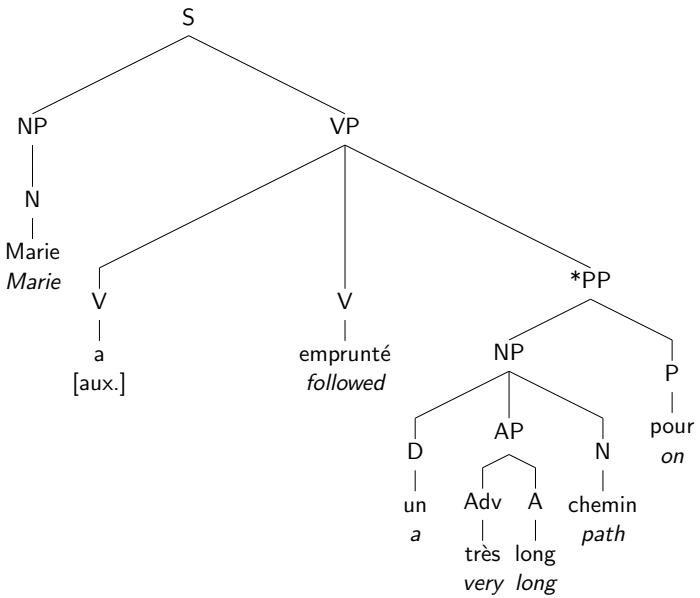
Optimality

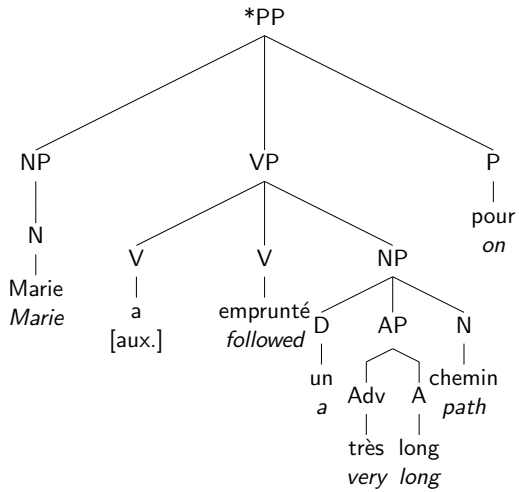
*Marie a emprunté un très long chemin pour
*Marie [aux.] took a very long path on

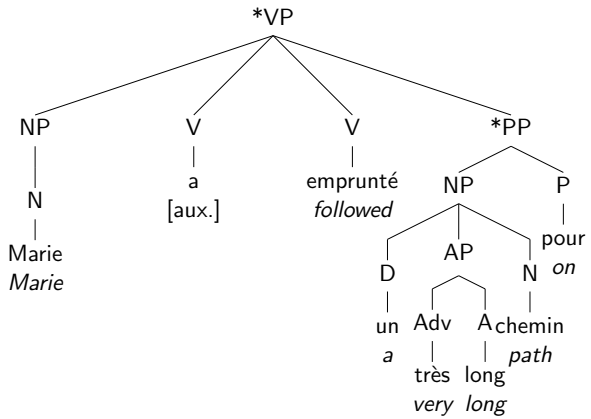
Intersective Gradience

Optimality

*Marie a emprunté un très long chemin pour
**Marie [aux.] took a very long path on*



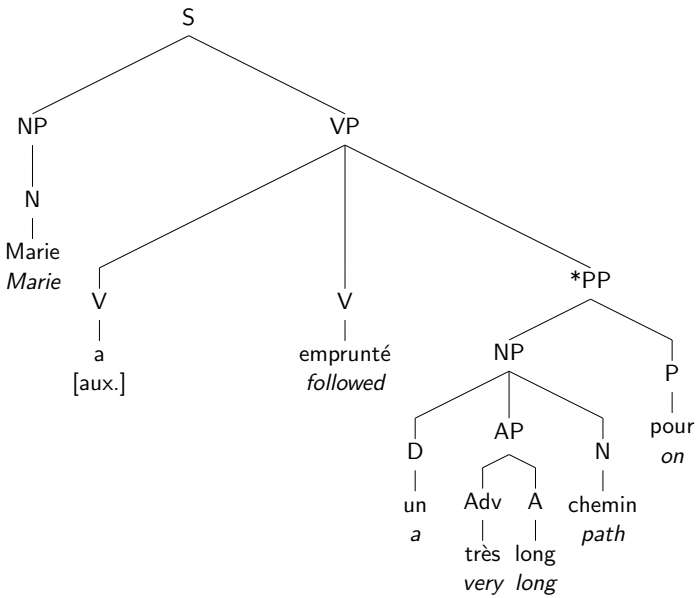




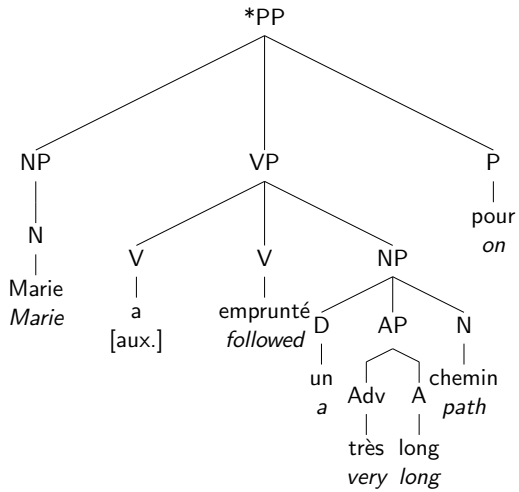
Intersective Gradience

Optimality

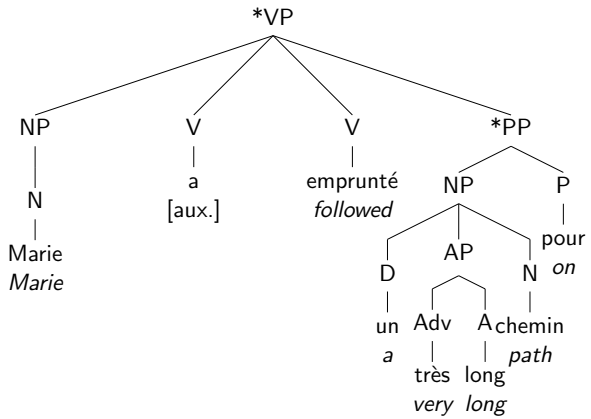
*Marie a emprunté un très long chemin pour
**Marie [aux.] took a very long path on*



merit



merit

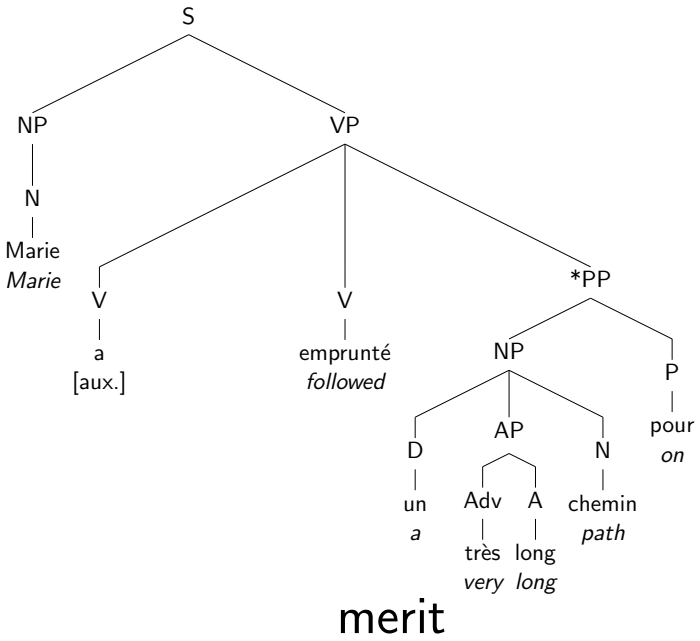


merit

Intersective Gradience

Optimality

*Marie a emprunté un très long chemin pour
**Marie [aux.] took a very long path on*



merit

Linear Optimality Theory

Keller, 2000

- Based on Optimality Theory (Prince and Smolenski, 1993)

Linear Optimality Theory

Keller, 2000

- Based on Optimality Theory (Prince and Smolenski, 1993)
- Ranks sub-optimal structures

Linear Optimality Theory

Keller, 2000

- Based on Optimality Theory (Prince and Smolenski, 1993)
- Ranks sub-optimal structures
- Optimality-Theoretic Grammaticality

Linear Optimality Theory

Keller, 2000

- Based on Optimality Theory (Prince and Smolenski, 1993)
- Ranks sub-optimal structures
- Optimality-Theoretic Grammaticality
  - Grammatical structure = optimal structure

Linear Optimality Theory

Keller, 2000

- Based on Optimality Theory (Prince and Smolenski, 1993)
- Ranks sub-optimal structures
- Optimality-Theoretic Grammaticality
  - Grammatical structure = optimal structure
  - Grammaticality *vs.* ungrammaticality?

Subsective Gradience

Rating

Marie a emprunté un très long chemin pour le retour	**4**
'Marie took a very long path on the way back'	
*Marie a emprunté emprunté un très long chemin pour le retour	3.9
**'Marie took took a very long path on the way back'*	
*Marie a emprunté un très long chemin pour	2.7
**'Marie took a very long path on'*	
*Marie un très long chemin pour le retour	1.3
**'Marie a very long path on the way back'*	

Classes: Construction

- English Caused-Motion Construction

Chloe sneezed the tissue off the table
'Chloé a éternué le mouchoir hors de la table'

- Phrases: NP, VP, D, A

'le juge', 'a octroyé', 'un', 'bref'

- Lexical Constructions: words

Construction Specification

Property Grammars (Blache, 2001)

S (*Utterance*)		
Feat.	Prop. Type : Properties	
[AVM]	*obligation* : $\triangle\mathrm{VP}$	(P1)
	uniqueness : $\mathrm{NP}!$	(P2)
	linearity : $\mathrm{NP} \prec \mathrm{VP}$	(P3)
	dependency : $\mathrm{NP} \rightsquigarrow \mathrm{VP}$	(P4)

NP (*Noun Phrase*)		
Feat.	Prop. Type : Properties	
$\begin{bmatrix}\textsc{gend}\\ \textsc{num}\end{bmatrix}$	*obligation* : $\mathit{Obl}(\mathrm{N} \vee \textsc{Pro})$	(P5)
	uniqueness : $\mathrm{D}!$	(P6)
	linearity : $\mathrm{D} \prec \mathrm{N}$	(P7)
	requirement : $\mathrm{N} \Rightarrow \mathrm{D}$	(P8)
	exclusion : $\mathrm{N} \Leftrightarrow \textsc{Pro}$	(P9)
	dependency : $\mathrm{N}\begin{bmatrix}\textsc{gend}\ \boxed{1}\\ \textsc{num}\ \boxed{2}\end{bmatrix} \rightsquigarrow \mathrm{D}\begin{bmatrix}\textsc{gend}\ \boxed{1}\\ \textsc{num}\ \boxed{2}\end{bmatrix}$	(P10)

Modelling Syntactic Gradience

Proposed Solution

- Characterise a well formed or ill formed sentence

Modelling Syntactic Gradience

Proposed Solution

- Characterise a well formed or ill formed sentence
- Generate an optimal full parse

Modelling Syntactic Gradience

Proposed Solution

- Characterise a well formed or ill formed sentence
- Generate an optimal full parse
- Rate the syntactic acceptability of the utterance

Outline

Model

Intuitively, a constituent is a model of an utterance, which satisfies the grammar

Model-Theoretic Syntax (MTS)

- Domain: constituents
- Grammar: set of pairs
  - Constraint: a well-formed formula ϕ in FOL

Example

$$\begin{aligned} \textit{obligation} &: \triangle \mathrm{VP} \\ \textit{linearity} &: \mathrm{NP} \prec \mathrm{VP} \end{aligned}$$

- Projection Rule:

$$r.\textsc{cat} = C \rightarrow \phi$$

Example

$$r.\textsc{cat} = \mathrm{NP} \rightarrow \mathsf{obl}(r) \wedge \mathsf{lin}(r)$$

Constraint Satisfaction

Definition (Strict Satisfaction)

$$\mathcal{M} \models \bigwedge_{i \in \{1,\ldots,n\}} \phi_i$$

Constraint Satisfaction

Definition (Strict Satisfaction)

$$\mathcal{M} \models \bigwedge_{i \in \{1,\ldots,n\}} \phi_i$$

Definition (Loose Satisfaction)

$$\mathcal{M} \models\!\!\!\sim \bigwedge_{i \in \{1,\ldots,n\}} \phi_i$$

Constraint Satisfaction

Definition (Strict Satisfaction)

$$\mathcal{M} \models \bigwedge_{i \in \{1,\ldots,n\}} \phi_i$$

Definition (Loose Satisfaction)

$$\mathcal{M} \models\!\!\!\sim \bigwedge_{i \in \{1,\ldots,n\}} \phi_i$$

$$\mathcal{M} \models \bigwedge_{i \in \{1,\ldots,n\} \setminus I_k} \phi_i \wedge \bigwedge_{j \in I_k} \neg \psi_j$$

Parsing

Characterisation and Optimality

- Loose Constraint Satisfaction
- Dynamic Programming (CKY skeleton)
  - Chart (Dynamic Programming Table)
  - Memoization

Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

Example

*Le juge octroie bref entretien à ce plaignant

**'The judge grants brief interview to this plaintiff'*

Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

Example

*Le juge octroie bref entretien à ce plaignant

**'The judge grants brief interview to this plaintiff'*

Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

Example

*__Le__ juge octroie bref entretien à ce plaignant

**'The judge grants brief interview to this plaintiff'*

Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

Example

*__Le juge__ octroie bref entretien à ce plaignant

**‘The judge grants brief interview to this plaintiff’*

Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

Example

*__Le juge octroie__ bref entretien à ce plaignant

**‘The judge grants brief interview to this plaintiff’*

Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

Example

*__Le juge octroie__ bref entretien à ce plaignant

**‘The judge grants brief interview to this plaintiff’*

Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

Example

*Le **juge octroie bref** entretien à ce plaignant

**'The judge grants brief interview to this plaintiff'*

Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

Example

*Le juge **octroie bref entretien** à ce plaignant

**'The judge grants brief interview to this plaintiff'*

Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

Example

*Le juge octroie **bref entretien à** ce plaignant

**'The judge grants brief interview to this plaintiff'*

Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

Example

*Le juge octroie **bref entretien à** ce plaignant

**'The judge grants brief interview to this plaintiff'*

Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

Example

*Le juge octroie **bref entretien à** ce plaignant

**'The judge grants brief interview to this plaintiff'*

Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

Example

*Le juge octroie **bref entretien à** ce plaignant

**'The judge grants brief interview to this plaintiff'*

Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

Example

*Le juge octroie **bref entretien à** ce plaignant

**'The judge grants brief interview to this plaintiff'*

Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

Example

*Le juge octroie **bref entretien à** ce plaignant

**'The judge grants brief interview to this plaintiff'*

LSCP

Walkthrough

Example

*Le	juge	octroie	bref	entretien	à	ce	plaignant
**The*	*judge*	*grants*	*brief*	*interview*	*to*	*this*	*plaintiff*

LSCP

Walkthrough

Example

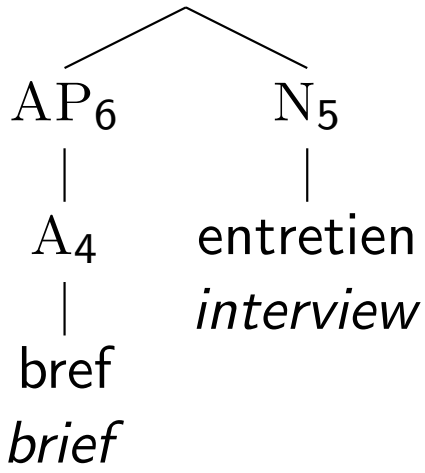
*Le	juge	octroie	**bref**	**entretien**	à	ce	plaignant
**The*	*judge*	*grants*	*brief*	*interview*	*to*	*this*	*plaintiff*

LSCP

Walkthrough

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

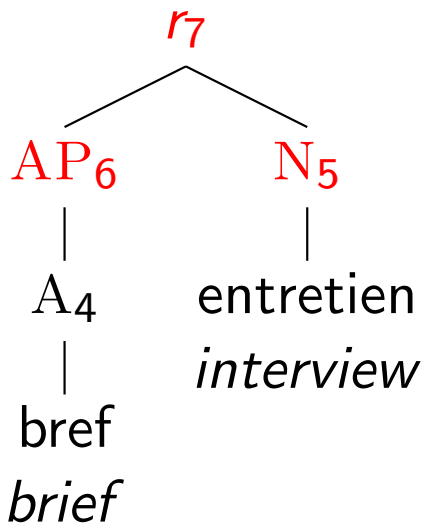


LSCP

Walkthrough

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

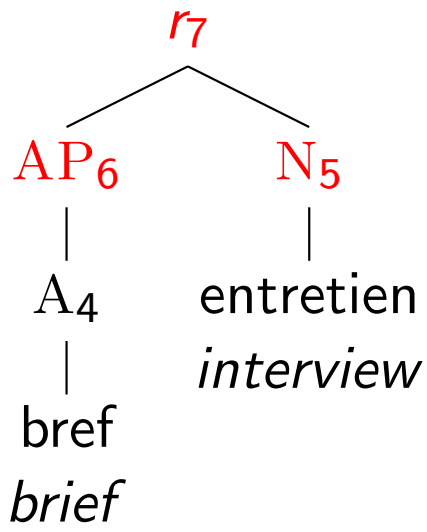


LSCP

Walkthrough

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```



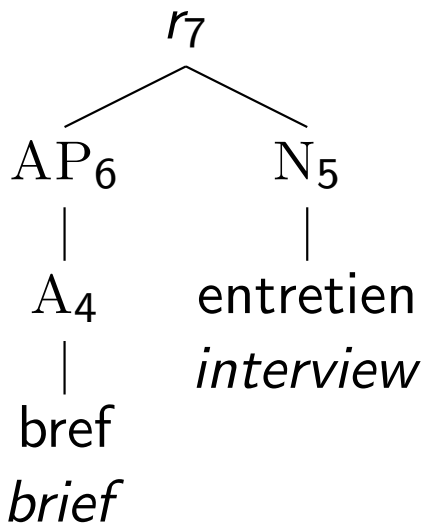
$$\mathcal{A} = \langle \|r_7\| = \mathrm{R}_7, \mathrm{AP}_6, \mathrm{N}_5 \rangle$$

LSCP

Walkthrough

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```



$$\mathcal{A} = \langle \|r_7\| = \mathrm{R}_7, \mathrm{AP}_6, \mathrm{N}_5 \rangle$$

Grammar Lookup

NP (*Noun Phrase*)

Feat.	Prop. Type : Properties	
[AVM]	*obligation* : $Obl(\mathrm{N} \vee \textsc{Pro})$	(P11)
	uniqueness : $\mathrm{D}!$	(P12)
	: $\mathrm{N}!$	(P13)
	: $\mathrm{PP}!$	(P14)
	: $\textsc{Pro}!$	(P15)
	linearity : $\mathrm{D} \prec \mathrm{N}$	(P16)
	: $\mathrm{D} \prec \textsc{Pro}$	(P17)
	: $\mathrm{D} \prec \mathrm{AP}$	(P18)
	: $\mathrm{N} \prec \mathrm{PP}$	(P19)
	requirement : $\mathrm{N} \Rightarrow \mathrm{D}$	(P20)
	: $\mathrm{AP} \Rightarrow \mathrm{N}$	(P21)
	exclusion : $\mathrm{N} \nLeftrightarrow \textsc{Pro}$	(P22)
	dependency : $\mathrm{N}\left[\begin{smallmatrix}\textsc{gend}\ \boxed{1}\\ \textsc{num}\ \boxed{2}\end{smallmatrix}\right] \rightsquigarrow \mathrm{D}\left[\begin{smallmatrix}\textsc{gend}\ \boxed{1}\\ \textsc{num}\ \boxed{2}\end{smallmatrix}\right]$	(P23)

VP (*Verb Phrase*)

Feat.	Prop. Type : Properties	
[AVM]	*obl.* : $\triangle \mathrm{V}$	(P24)
	uniq. : $\mathrm{V}_{[\textit{main past part.}]}!$	(P25)
	: $\mathrm{NP}!$	(P26)
	: $\mathrm{PP}!$	(P27)
	lin. : $\mathrm{V} \prec \mathrm{NP}$	(P28)
	: $\mathrm{V} \prec \textsc{Adv}$	(P29)
	: $\mathrm{V} \prec \mathrm{PP}$	(P30)
	req. : $\mathrm{V}_{[\textit{past part.}]} \Rightarrow \mathrm{V}_{[\textit{aux.}]}$	(P31)
	excl. : $\textsc{Pro}_{[\textit{acc}]} \nLeftrightarrow \mathrm{NP}$	(P32)
	: $\textsc{Pro}_{[\textit{dat}]} \nLeftrightarrow \textsc{Pro}_{[\textit{acc}]}$	(P33)
	dep. : $\mathrm{V}\left[\begin{smallmatrix}\textsc{pers}\ \boxed{1}\\ \textsc{num}\ \boxed{2}\end{smallmatrix}\right] \rightsquigarrow \textsc{Pro}\left[\begin{smallmatrix}\textsc{type}\ \textit{pers}\\ \textsc{case}\ \textit{nom}\\ \textsc{pers}\ \boxed{1}\\ \textsc{num}\ \boxed{2}\end{smallmatrix}\right]$	(P34)

S (*Utterance*)

Feat.	Prop. Type : Properties	
[AVM]	*obl.* : $\triangle \mathrm{VP}$	(P35)
	uniq. : $\mathrm{NP}!$	(P36)
	: $\mathrm{VP}!$	(P37)
	lin. : $\mathrm{NP} \prec \mathrm{VP}$	(P38)
	dep. : $\mathrm{NP} \rightsquigarrow \mathrm{VP}$	(P39)

AP (*Adjective Phrase*)

Feat.	Prop. Type : Properties	
[AVM]	*obl.* : $Obl(\mathrm{A} \vee \mathrm{V}_{[\textit{past part.}]})$	(P40)
	uniq. : $\mathrm{A}!$	(P41)
	: $\mathrm{V}_{[\textit{past part.}]}!$	(P42)
	: $\textsc{Adv}!$	(P43)
	lin. : $\mathrm{A} \prec \mathrm{PP}$	(P44)
	: $\textsc{Adv} \prec \mathrm{A}$	(P45)
	excl. : $\mathrm{A} \nLeftrightarrow \mathrm{V}_{[\textit{past part.}]}$	(P46)

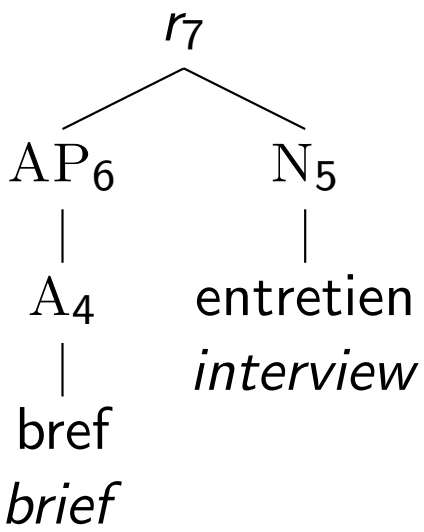
PP (*Prepositional Phrase*)

Feat.	Prop. Type : Properties	
[AVM]	*obl.* : $\triangle \mathrm{P}$	(P47)
	uniq. : $\mathrm{P}!$	(P48)
	: $\mathrm{NP}!$	(P49)
	lin. : $\mathrm{P} \prec \mathrm{NP}$	(P50)
	: $\mathrm{P} \prec \mathrm{VP}$	(P51)
	req. : $\mathrm{P} \Rightarrow \mathrm{NP}$	(P52)
	dep. : $\mathrm{P} \rightsquigarrow \mathrm{NP}$	(P53)

LSCP Walkthrough

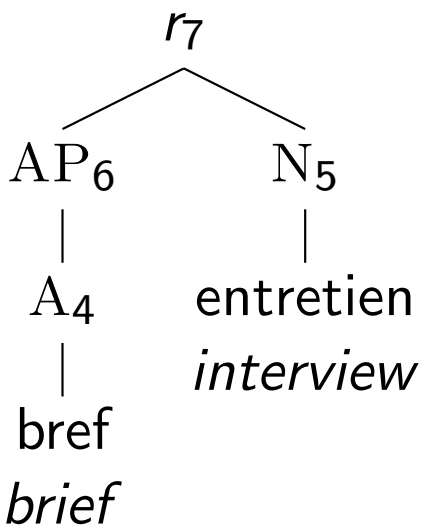
Characterisation

$$\mathcal{A} = \langle \|r_7\| = \mathrm{R}_7, \mathrm{AP}_6, \mathrm{N}_5 \rangle$$



LSCP Walkthrough

Characterisation



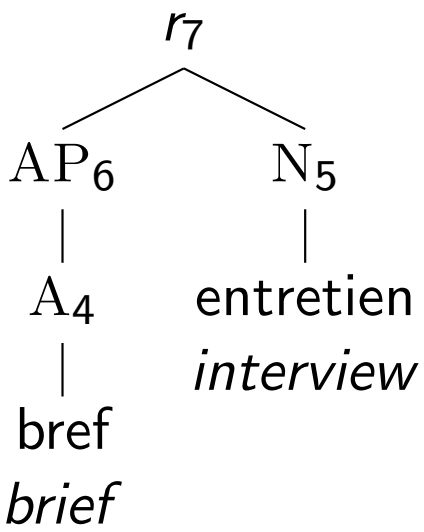
$$\mathcal{A} = \langle \|r_7\| = \mathrm{R}_7, \mathrm{AP}_6, \mathrm{N}_5 \rangle$$

$$\mathcal{A} \models (\mathrm{P11}) \wedge (\mathrm{P21}) \wedge (\mathrm{P22}) \quad (1)$$

$$:= \chi^+_{\mathcal{A}} \quad (2)$$

LSCP Walkthrough

Characterisation



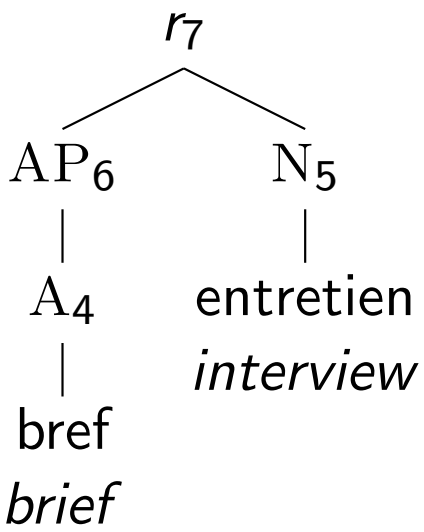
$$\mathcal{A} = \langle \|r_7\| = \mathrm{R}_7, \mathrm{AP}_6, \mathrm{N}_5 \rangle$$

$$\mathcal{A} \models (\mathrm{P11}) \wedge (\mathrm{P21}) \wedge (\mathrm{P22}) := \chi^+_{\mathcal{A}} \quad (1)$$

$$\mathcal{A} \not\models (\mathrm{P20}) := \chi^-_{\mathcal{A}} \quad (2)$$

LSCP Walkthrough

Characterisation



$$\mathcal{A} = \langle \|r_7\| = \mathrm{R}_7, \mathrm{AP}_6, \mathrm{N}_5 \rangle$$

$$\mathcal{A} \models (\mathrm{P11}) \wedge (\mathrm{P21}) \wedge (\mathrm{P22}) := \chi^+_{\mathcal{A}} \quad (1)$$

$$\mathcal{A} \not\models (\mathrm{P20}) := \chi^-_{\mathcal{A}} \quad (2)$$

$$\mathcal{A} \models\!\!\backsim \chi^+_{\mathcal{A}} \wedge \chi^-_{\mathcal{A}}$$

LSCP Walkthrough

Projection

Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

$$\mathcal{A} \models \mathrm{NP}_7.\textsc{cat} = \mathbf{NP} \rightarrow \chi^{+}_{\mathcal{A}} \wedge \chi^{-}_{\mathcal{A}}$$

LSCP Walkthrough

Projection

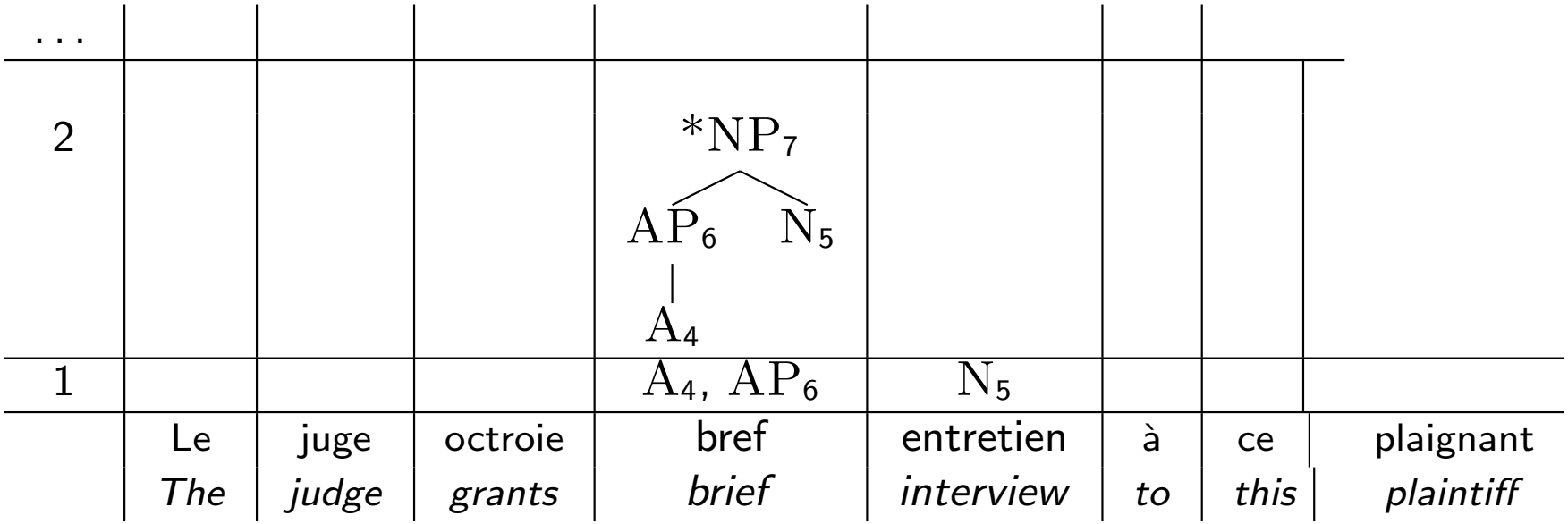
Algorithm

```
for (span=1 to num_words)
  for (offset=1 to num_words-span+1)
    for (every assignment A in [offset..end])
      X ← characterisation(A)
      C ← projection(X)
      for (every x in C)
        if (merit(x) >= pi[offset, span, C]) then
          pi[offset, span, C] ← {x,merit(x)}
```

$$\mathcal{A} \models \text{NP}_7.\text{CAT} = \textbf{NP} \rightarrow \chi^+_{\mathcal{A}} \wedge \chi^-_{\mathcal{A}}$$

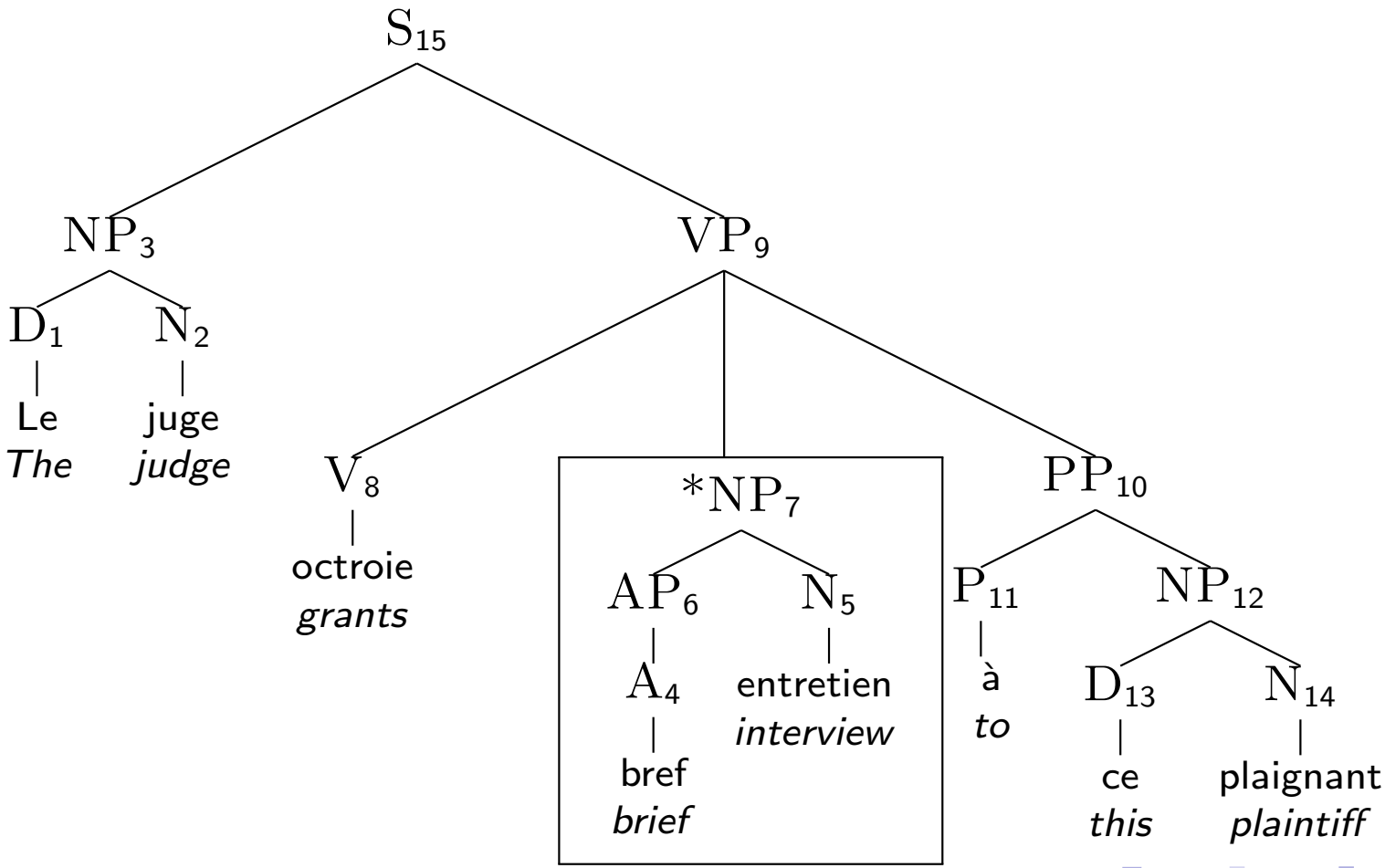
LSCP Walkthrough

Memoization



LSCP

Solution Parse



Evaluation of *Numbat*

EASY

	Precision	Recall	F_1
Numbat	**78.4%**	**70.6%**	**74.2%**

Evaluation of *Numbat*

EASY

	Precision	Recall	F_1
Numbat	**78.4%**	**70.6%**	**74.2%**
LPL Shallow parser	78.5%	83.8%	81 %
LPL stochastic parser	90.1 %	89.8%	89.9%

Evaluation of *Numbat*

EASY

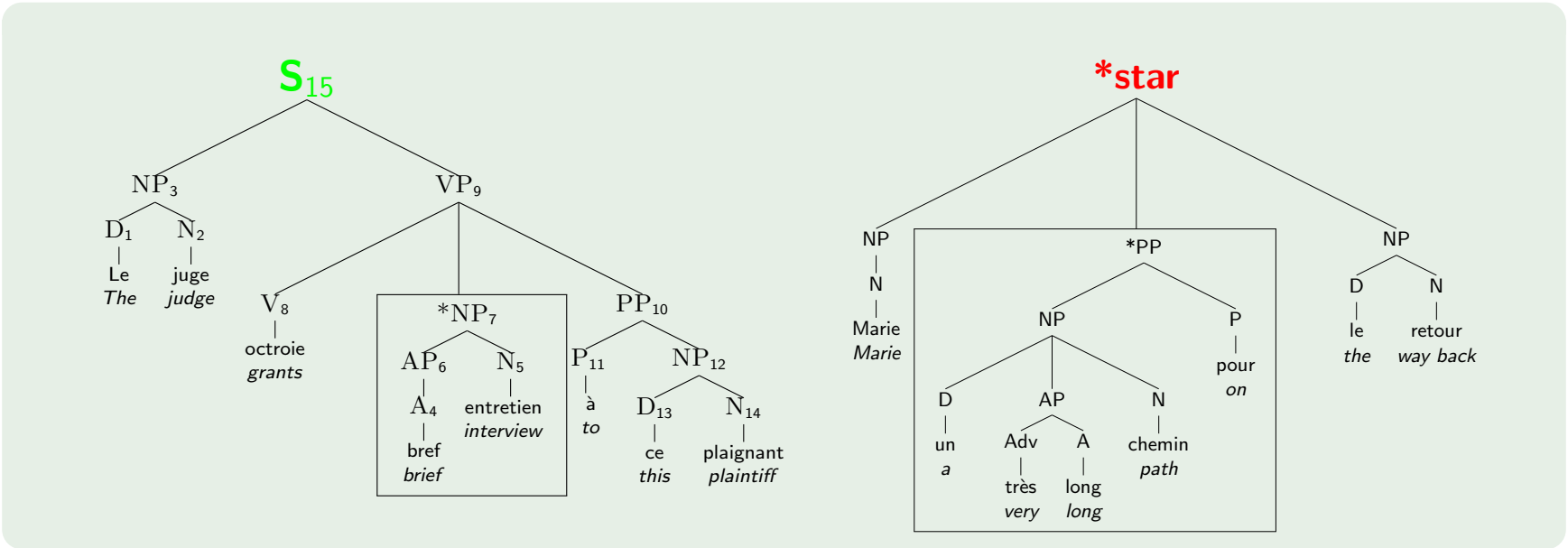
Example

j'entendais	encore,	au fond	du pavillon,	monsieur	qui	essayait	d'ébranler	la porte.
NV	GR	GP	GP	GN	GN	NV	PV	GN
VP	*AdvP*	*PP*	*PP*	*NP*	*NP*	*VP*	*Prep. VP*	*NP*

'I was still hearing, at the back of the house, Sir, who was trying to shake the door.'

Evaluation of *Numbat*

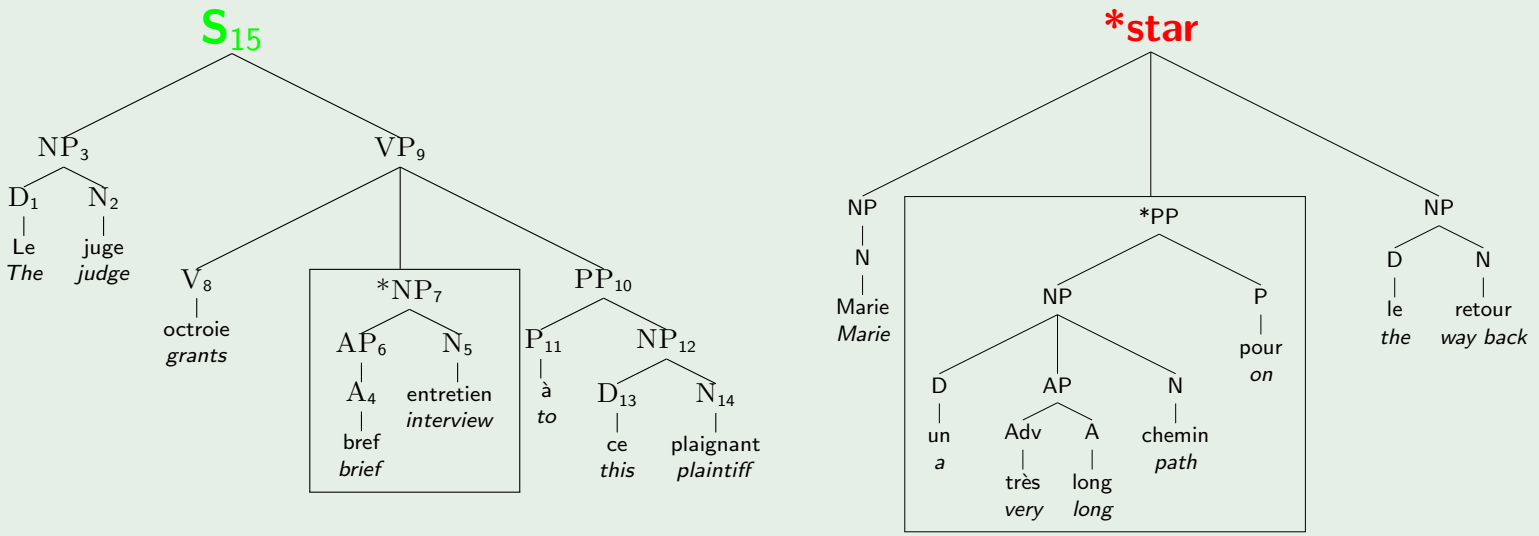
Quasi-expressions



Evaluation of *Numbat*

Quasi-expressions

	Precision $\frac{correct}{complete}$	Recall $\frac{correct}{total}$	F_1
Numbat	**74%**	**68%**	**71%**



Summary

- Automated syntactic parsing (LSCP)
- Well formed or ill formed utterance
- Intersective Gradience:
  - Optimal parse (constituent structure)
  - Characterisation of maximum merit for a given grammar

Outline

Problem

Subsective Gradience

Can we predict how grammatically acceptable an utterance is?

Central Claim

The degree of acceptability of an utterance can be predicted by factors derivable from the outcome of LSCP

Predicting Acceptability

Postulates

1. Failure Cumulativity
   - Violated constraints
   - N_c^-

Predicting Acceptability

Postulates

1. Failure Cumulativity
   - Violated constraints
   - N_c^-
2. Success Cumulativity
   - Satisfied constraints
   - N_c^+

Postulates

3 Constraint Weighting

- $W_c^+ = \sum_c w^+$
- $W_c^- = \sum_c w^-$

Postulates

3 Constraint Weighting
- $W_c^+ = \sum_c w^+$
- $W_c^- = \sum_c w^-$

4 Constructional Complexity
- $T_{\mathcal{C}}$: Total number of constraints specifying the construction $\mathcal{C}$
- E_c: Total number of constraints evaluated for the constituent c

Postulates

- Propagation
  - Acceptability of the whole depends on acceptability of the parts
  - $f(c) = k \cdot f(c_i)$

Predicting Acceptability

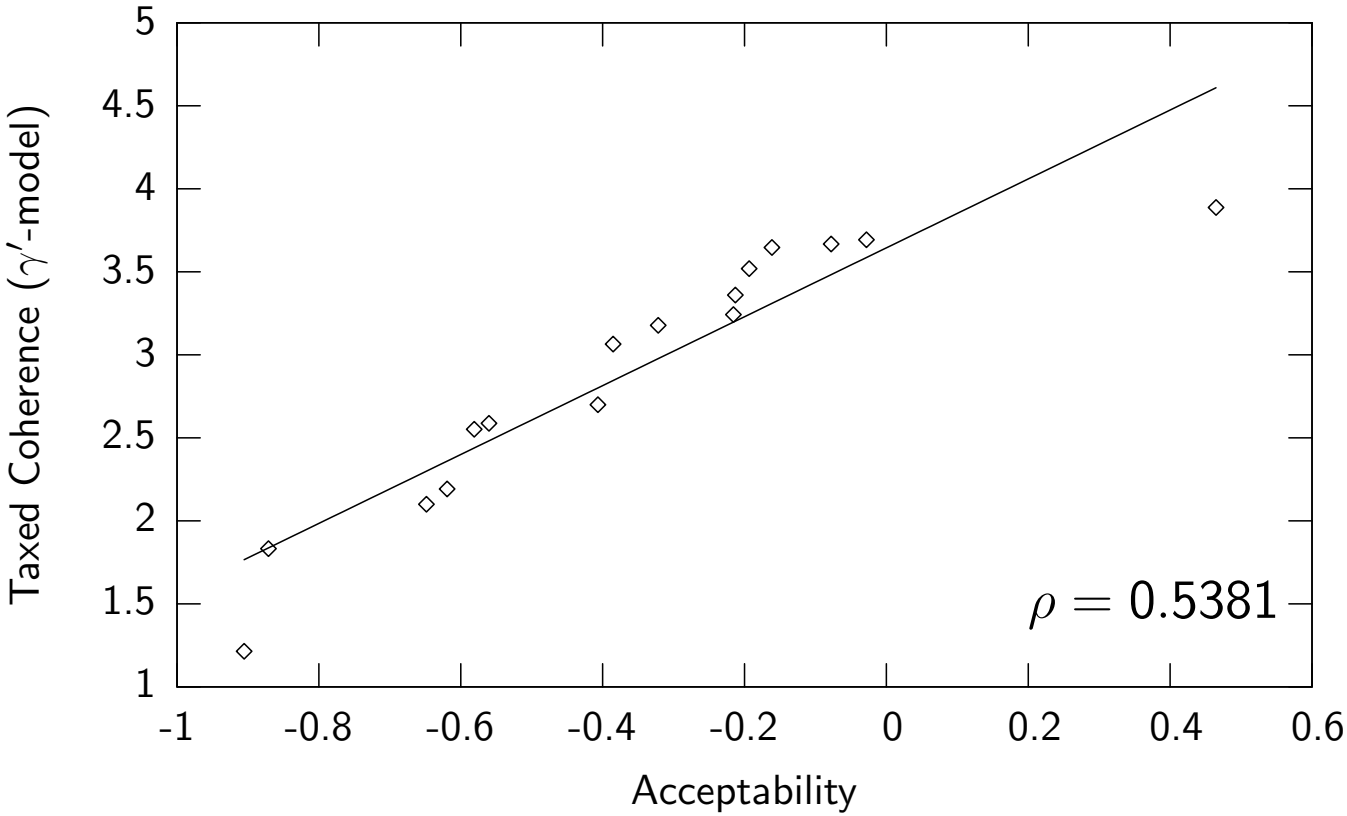
Numerical Models

- Cohesion (γ)
- Taxed Cohesion (γ')

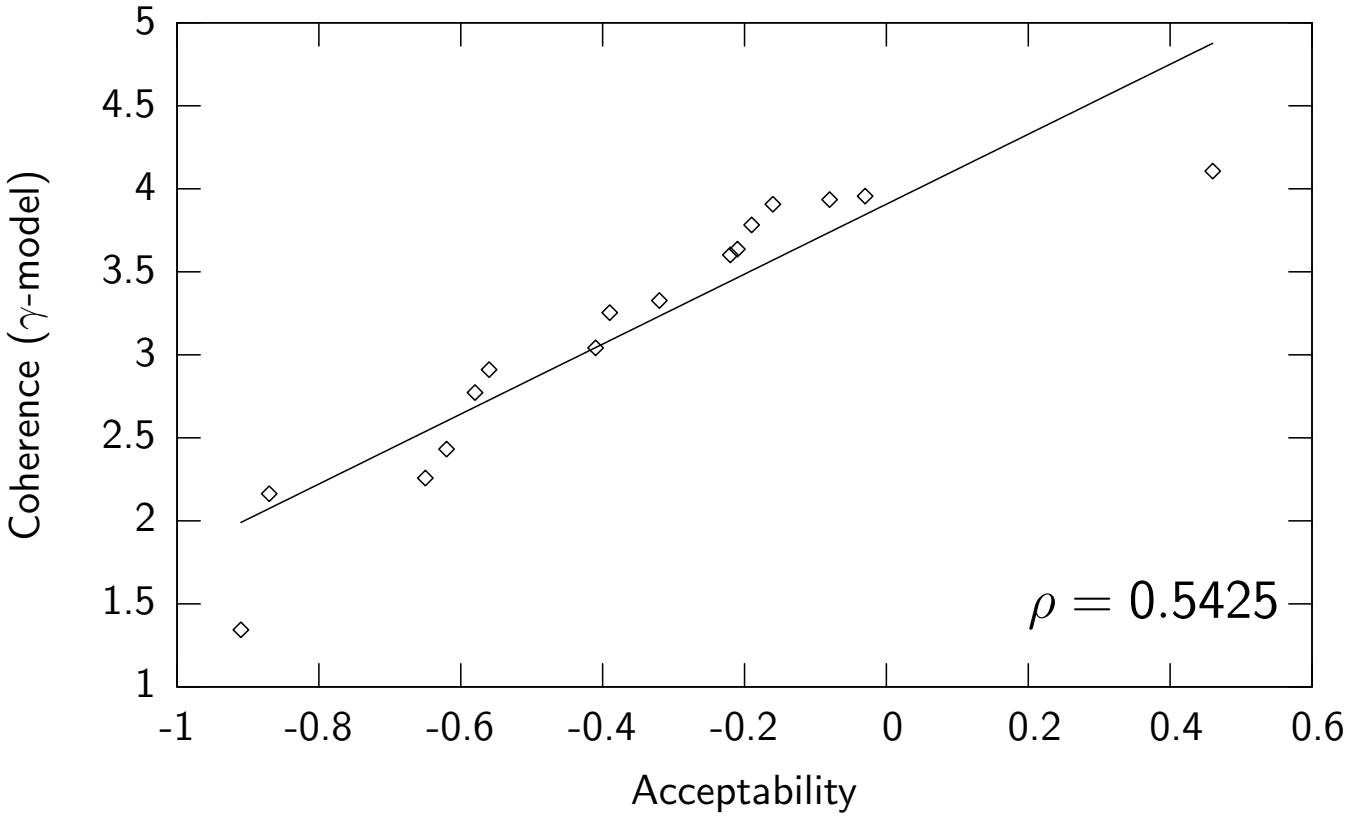
Human Judgements by Error Patterns

No violations		
1.1	Marie a emprunté un très long chemin pour le retour *Marie [aux.] followed a very long path on the way back*	0.465
NP-violations		
2.1	Marie a emprunté très long chemin un pour le retour *Marie [aux.] followed very long path a on the way back*	-0.643
...		
AP-violations		
3.2	Marie a emprunté un très long long chemin pour le retour *Marie [aux.] followed a very long long path on the way back*	-0.216
...		
PP-violations		
4.3	Marie a emprunté un très long chemin le retour *Marie [aux.] followed a very long path the way back*	-0.213
...		
VP-violations		
5.4	Marie emprunté un très long chemin pour le retour *followed a very long path on the way back*	-0.322
...		

Model Fit


Taxed Coherence (γ'-model)
Acceptability
$\rho = 0.5381$
5
4.5
4
3.5
3
2.5
2
1.5
1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6

Model Fit


Coherence (γ-model)
Acceptability
$\rho = 0.5425$
5
4.5
4
3.5
3
2.5
2
1.5
1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6

Scale of γ–scores

Type	2.3	1.1	5.2	3.2	4.2	3.1	2.2	4.3
γ	4.11	3.96	3.93	3.91	3.78	3.64	3.60	3.33
Ref. Rank	15	1	5	7	3	10	4	6
Rank	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**

Type	2.4	3.3	5.4	4.1	2.1	5.1	4.4	5.3
γ	3.25	3.042	2.91	2.77	2.43	2.26	2.16	1.34
Ref. Rank	2	13	8	12	14	11	9	16
Rank	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**

Scale of γ–scores

Type	2.3	1.1	5.2	3.2	4.2	3.1	2.2	4.3
γ	4.11	3.96	3.93	3.91	3.78	3.64	3.60	3.33
Ref. Rank	15	1	5	7	3	10	4	6
Rank	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**

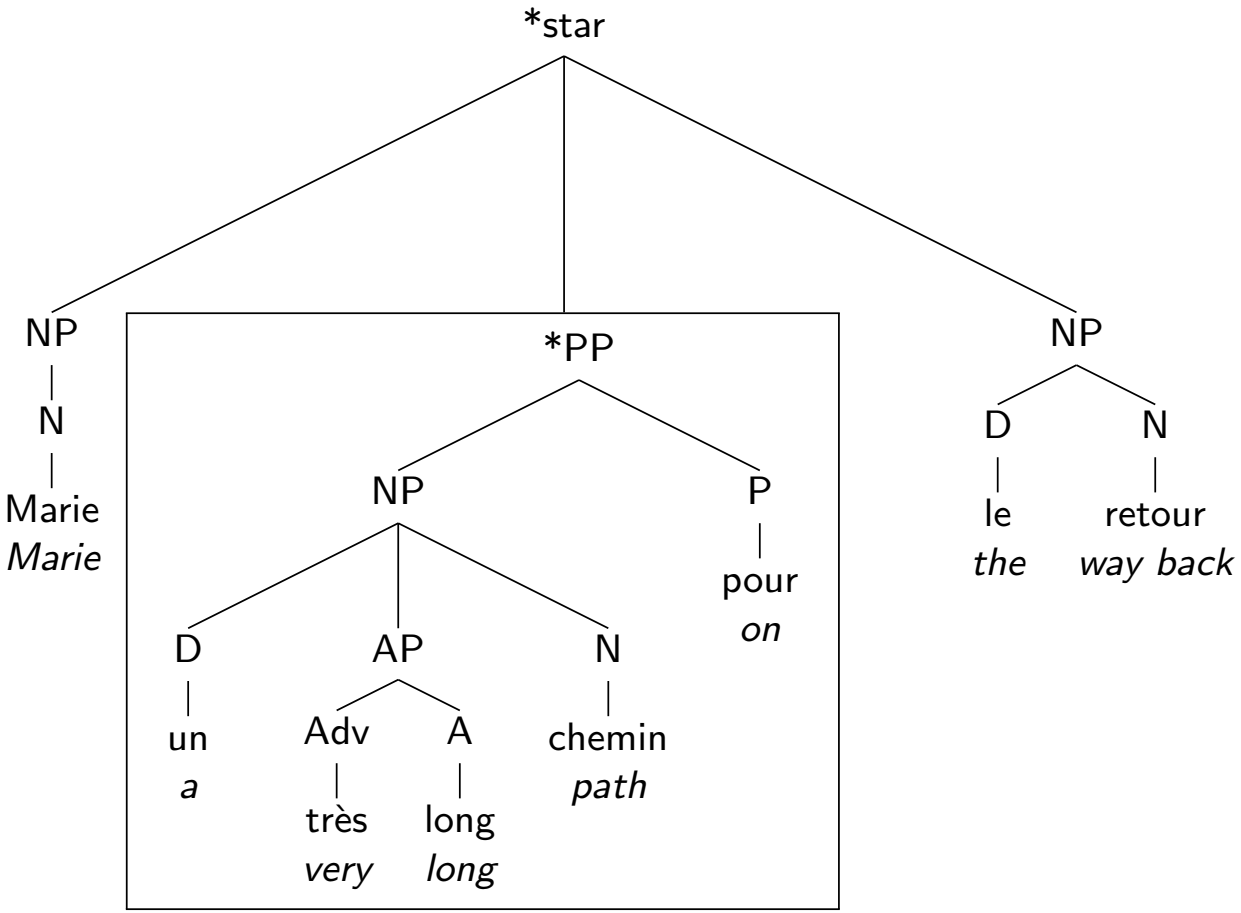
Type	2.4	3.3	5.4	4.1	2.1	5.1	4.4	5.3
γ	3.25	3.042	2.91	2.77	2.43	2.26	2.16	1.34
Ref. Rank	2	13	8	12	14	11	9	16
Rank	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**

Scale of γ–scores

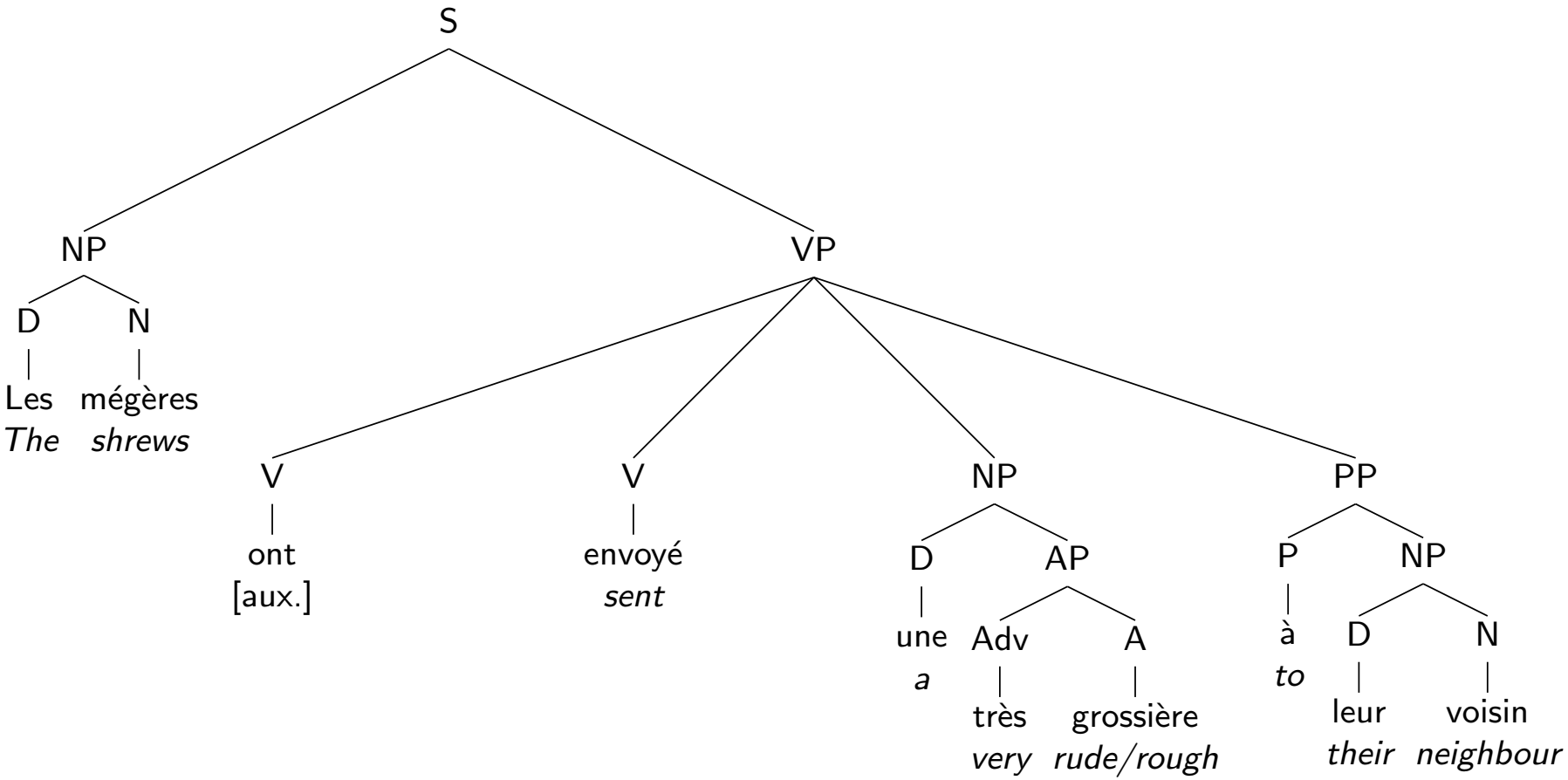
Type	2.3	1.1	5.2	3.2	4.2	3.1	2.2	4.3
γ	4.11	3.96	3.93	3.91	3.78	3.64	3.60	3.33
Ref. Rank	15	1	5	7	3	10	4	6
Rank	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**

Type	2.4	3.3	5.4	4.1	2.1	5.1	4.4	5.3
γ	3.25	3.042	2.91	2.77	2.43	2.26	2.16	1.34
Ref. Rank	2	13	8	12	14	11	9	16
Rank	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**

Missing VP (5.3)



Missing NP (2.3)



Ranking

Rank diff.	Occurrences		% of total
≥ 6	3		18.75%
$= 4$	1	6.25%	
≤ 3	12	75%	81.25%

Outline

Conclusion

- Model of syntactic gradience

Conclusion

- Model of syntactic gradience
  - Extended IG/SG

Conclusion

- Model of syntactic gradience
  - Extended IG/SG
    - well formed and ill formed language
    - at the constructional level

Conclusion

- Model of syntactic gradience
  - Extended IG/SG
    - well formed and ill formed language
    - at the constructional level
  - Intersective Gradience: Optimality

Conclusion

- Model of syntactic gradience
  - Extended IG/SG
    - well formed and ill formed language
    - at the constructional level
  - Intersective Gradience: Optimality
  - Subsective Gradience: Rating

Conclusion

- Automated solution

Conclusion

- Automated solution
  - Characterises a well formed or ill formed sentence

Conclusion

- Automated solution
  - Characterises a well formed or ill formed sentence
  - Generates an optimal full parse

Conclusion

- Automated solution
  - Characterises a well formed or ill formed sentence
  - Generates an optimal full parse
  - Predicts the utterance's syntactic acceptability by rating it

Conclusion

Contribution

- Specification of a basic Model-Theoretic framework for PG
- Implementation of LSCP
- Investigation of numeric models for predicting acceptability

Further Works

- Scaling up the model
  - To a large-coverage grammar
  - To non-artificial sentences
- Generalisation of constraint-based parsing to configuration

The End

Thank You