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)
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

1 Background
   • Gradience
   • Modelling Gradience
   • Classification

2 Intersective Gradience and Optimality
   • Knowledge Representation
   • Loose Satisfaction Chart Parsing
   • Evaluation

3 Subsective Gradience and Acceptability
   • Predicting Acceptability
   • Empirical Investigation
   • Interpretation

4 Conclusion
1 Background
   • Gradience
   • Modelling Gradience
   • Classification

2 Intersective Gradience and Optimality
   • Knowledge Representation
   • Loose Satisfaction Chart Parsing
   • Evaluation

3 Subsective Gradience and Acceptability
   • Predicting Acceptability
   • Empirical Investigation
   • Interpretation

4 Conclusion
Outline

1. Background
   - Gradience
   - Modelling Gradience
   - Classification

2. Intersective Gradience and Optimality
   - Knowledge Representation
   - Loose Satisfaction Chart Parsing
   - Evaluation

3. Subsective Gradience and Acceptability
   - Predicting Acceptability
   - Empirical Investigation
   - Interpretation

4. Conclusion
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 and Optimality

*Marie a emprunté un très long chemin pour

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

<table>
<thead>
<tr>
<th>French Phrase</th>
<th>English Translation</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marie a emprunté un très long chemin pour le retour</td>
<td>‘Marie took a very long path on the way back’</td>
<td>4</td>
</tr>
<tr>
<td>*Marie a emprunté emprunté un très long chemin pour le retour</td>
<td>*‘Marie took took a very long path on the way back’</td>
<td>3.9</td>
</tr>
<tr>
<td>*Marie a emprunté un très long chemin pour</td>
<td>*‘Marie took a very long path on’</td>
<td>2.7</td>
</tr>
<tr>
<td>*Marie un très long chemin pour le retour</td>
<td>*‘Marie a very long path on the way back’</td>
<td>1.3</td>
</tr>
</tbody>
</table>
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

- Lexical Constructions: words
  
  'le juge', 'a octroyé', 'un', 'bref'
# Construction Specification

## Property Grammars (Blache, 2001)

**S (Utterance)**

<table>
<thead>
<tr>
<th>Feat.</th>
<th>Prop. Type: Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>[AVM]</td>
<td>obligation: $\Delta VP$ (P1)</td>
</tr>
<tr>
<td></td>
<td>uniqueness: NP! (P2)</td>
</tr>
<tr>
<td></td>
<td>linearity: NP $\prec VP$ (P3)</td>
</tr>
<tr>
<td></td>
<td>dependency: NP $\sim VP$ (P4)</td>
</tr>
</tbody>
</table>

**NP (Noun Phrase)**

<table>
<thead>
<tr>
<th>Feat.</th>
<th>Prop. Type: Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>[GEND][NUM]</td>
<td>obligation: $Obl(N \lor PRO)$ (P5)</td>
</tr>
<tr>
<td></td>
<td>uniqueness: D! (P6)</td>
</tr>
<tr>
<td></td>
<td>linearity: D $\prec N$ (P7)</td>
</tr>
<tr>
<td></td>
<td>requirement: $N \Rightarrow D$ (P8)</td>
</tr>
<tr>
<td></td>
<td>exclusion: $N \nRightarrow PRO$ (P9)</td>
</tr>
</tbody>
</table>
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

1 Background
   - Gradience
   - Modelling Gradience
   - Classification

2 Intersective Gradience and Optimality
   - Knowledge Representation
   - Loose Satisfaction Chart Parsing
   - Evaluation

3 Subsective Gradience and Acceptability
   - Predicting Acceptability
   - Empirical Investigation
   - Interpretation

4 Conclusion
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 $\phi$ in FOL

**Example**

<table>
<thead>
<tr>
<th>Obligation</th>
<th>$\Delta$VP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linearity</td>
<td>NP $\prec$ VP</td>
</tr>
</tbody>
</table>

**Projection Rule:**

$$r.\text{CAT} = C \rightarrow \phi$$

**Example**

$$r.\text{CAT} = \text{NP} \rightarrow \text{obl}(r) \land \text{lin}(r)$$
Definition (Strict Satisfaction)

\[ \mathcal{M} \models \bigwedge_{i \in \{1,\ldots,n\}} \phi_i \]
Constraint Satisfaction

**Definition (Strict Satisfaction)**

\[ M \models \bigwedge_{i \in \{1, \ldots, n\}} \phi_i \]

**Definition (Loose Satisfaction)**

\[ M \models^\prime \bigwedge_{i \in \{1, \ldots, n\}} \phi_i \]
Constraint Satisfaction

**Definition (Strict Satisfaction)**

\[ M \models \bigwedge_{i \in \{1, \ldots, n\}} \phi_i \]

**Definition (Loose Satisfaction)**

\[ M \models \bigwedge_{i \in \{1, \ldots, n\}} \phi_i \]

\[ M \models \bigwedge_{i \in \{1, \ldots, n\} \setminus l_k} \phi_i \land \bigwedge_{j \in l_k} \neg \psi_j \]
Parsing

Characterisation and Optimality

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

```plaintext
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

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

Example

*Le juge octroie bref entretien à ce plaignant*

*‘The judge grants brief interview to this plaintiff’*
Loose Satisfaction Chart Parsing (LSCP)

Algorithm

```plaintext
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

```plaintext
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

\[
\text{for (span=1 to num\_words)} \\
\quad \text{for (offset=1 to num\_words-span+1)} \\
\quad \quad \text{for (every assignment A in [offset..end])} \\
\quad \quad \quad \text{X} \leftarrow \text{characterisation(A)} \\
\quad \quad \quad \text{C} \leftarrow \text{projection(X)} \\
\quad \quad \text{for (every x in C)} \\
\quad \quad \quad \text{if (merit(x) >= pi[offset, span, C]) then} \\
\quad \quad \quad \quad \text{pi[offset, span, C] \leftarrow \{x, merit(x)\}}
\]

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’
**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’*
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])
      \( X \leftarrow \text{characterisation}(A) \)
      \( C \leftarrow \text{projection}(X) \)
      for (every \( x \) in \( C \))
        if (merit\((x) \geq pi[offset, span, C]) \) then
          \( pi[offset, span, C] \leftarrow \{x, \text{merit}(x)\} \)

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’
Example

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

*Le juge octroie bref entretien à ce plaignant

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

```plaintext
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)}
```

```
AP₆
  |   N₅
  |    entretien
  |    interview
A₄   brief
    brief
```
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)}

r7

AP6

N5

A4

entretien

bref

brief
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)}

\[ A = \langle \|r_7\| = R_7, AP_6, N_5 \rangle \]
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)}

\[ A = \langle \| r_7 \| = R_7, AP_6, N_5 \rangle \]
## Grammar Lookup

### NP (Noun Phrase)

<table>
<thead>
<tr>
<th>Prop. Type : Properties</th>
<th>Feature</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>obligation : $\text{Obli}(N \lor PRO)$</td>
<td>(P11)</td>
<td></td>
</tr>
<tr>
<td>uniqueness : $D!$</td>
<td>(P12)</td>
<td></td>
</tr>
<tr>
<td>: $N!$</td>
<td>(P13)</td>
<td></td>
</tr>
<tr>
<td>: $PP!$</td>
<td>(P14)</td>
<td></td>
</tr>
<tr>
<td>: $PRO!$</td>
<td>(P15)</td>
<td></td>
</tr>
<tr>
<td>linearity : $D \prec N$</td>
<td>(P16)</td>
<td></td>
</tr>
<tr>
<td>: $D \prec PRO$</td>
<td>(P17)</td>
<td></td>
</tr>
<tr>
<td>: $D \prec AP$</td>
<td>(P18)</td>
<td></td>
</tr>
<tr>
<td>: $N \prec PP$</td>
<td>(P19)</td>
<td></td>
</tr>
<tr>
<td>requirement : $N \Rightarrow D$</td>
<td>(P20)</td>
<td></td>
</tr>
<tr>
<td>: $AP \Rightarrow N$</td>
<td>(P21)</td>
<td></td>
</tr>
<tr>
<td>exclusion : $N \Leftrightarrow PRO$</td>
<td>(P22)</td>
<td></td>
</tr>
<tr>
<td>dependency : $N^{\text{GEND}[1]} \sim D^{\text{GEND}[1]}$</td>
<td>(P23)</td>
<td></td>
</tr>
</tbody>
</table>

### VP (Verb Phrase)

<table>
<thead>
<tr>
<th>Prop. Type : Properties</th>
<th>Feature</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>obligation : $\Delta V$</td>
<td>(P24)</td>
<td></td>
</tr>
<tr>
<td>uniqueness : $V_{\text{main past part}}!$</td>
<td>(P25)</td>
<td></td>
</tr>
<tr>
<td>: $NP!$</td>
<td>(P26)</td>
<td></td>
</tr>
<tr>
<td>: $PP!$</td>
<td>(P27)</td>
<td></td>
</tr>
<tr>
<td>lin. : $V \prec NP$</td>
<td>(P28)</td>
<td></td>
</tr>
<tr>
<td>: $V \prec ADV$</td>
<td>(P29)</td>
<td></td>
</tr>
<tr>
<td>: $V \prec PP$</td>
<td>(P30)</td>
<td></td>
</tr>
<tr>
<td>req. : $V_{\text{past part}} \Rightarrow V_{\text{aux}}$</td>
<td>(P31)</td>
<td></td>
</tr>
<tr>
<td>excl. : $\text{PRO}[\text{type}] \Rightarrow \text{PRO}[\text{case}]$</td>
<td>(P32)</td>
<td></td>
</tr>
<tr>
<td>dep. : $\text{PRO}[\text{type}] \sim \text{PRO}[\text{case}]$</td>
<td>(P33)</td>
<td></td>
</tr>
<tr>
<td>: $\text{PRO}[\text{pers}][\text{num}] \sim \text{PRO}[\text{pers}][\text{num}]$</td>
<td>(P34)</td>
<td></td>
</tr>
</tbody>
</table>

### AP (Adjective Phrase)

<table>
<thead>
<tr>
<th>Prop. Type : Properties</th>
<th>Feature</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>obligation : $\text{Obli}(A \lor V_{\text{past part}})$</td>
<td>(P40)</td>
<td></td>
</tr>
<tr>
<td>uniqueness : $A!$</td>
<td>(P41)</td>
<td></td>
</tr>
<tr>
<td>: $V_{\text{past part}}!$</td>
<td>(P42)</td>
<td></td>
</tr>
<tr>
<td>: $ADV!$</td>
<td>(P43)</td>
<td></td>
</tr>
<tr>
<td>lin. : $AP \prec PP$</td>
<td>(P44)</td>
<td></td>
</tr>
<tr>
<td>: $ADV \prec A$</td>
<td>(P45)</td>
<td></td>
</tr>
<tr>
<td>excl. : $A \Rightarrow V_{\text{past part}}$</td>
<td>(P46)</td>
<td></td>
</tr>
</tbody>
</table>

### PP (Prepositional Phrase)

<table>
<thead>
<tr>
<th>Prop. Type : Properties</th>
<th>Feature</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>obligation : $\Delta P$</td>
<td>(P47)</td>
<td></td>
</tr>
<tr>
<td>uniqueness : $PI!$</td>
<td>(P48)</td>
<td></td>
</tr>
<tr>
<td>: $NP!$</td>
<td>(P49)</td>
<td></td>
</tr>
<tr>
<td>lin. : $P \prec NP$</td>
<td>(P50)</td>
<td></td>
</tr>
<tr>
<td>: $P \prec VP$</td>
<td>(P51)</td>
<td></td>
</tr>
<tr>
<td>req. : $P \Rightarrow NP$</td>
<td>(P52)</td>
<td></td>
</tr>
<tr>
<td>excl. : $P \Rightarrow NP$</td>
<td>(P53)</td>
<td></td>
</tr>
</tbody>
</table>

Jean-Philippe Prost  
Modelling Syntactic Gradience with LSCP
LSCP Walkthrough

Characterisation

\[ \mathcal{A} = \langle \| r_7 \| = R_7, \text{AP}_6, \text{N}_5 \rangle \]
LSCP Walkthrough

Characterisation

\[ \mathcal{A} = \langle \rVert r_7 \rVert = R_7, AP_6, N_5 \rangle \]

\[ \mathcal{A} \models (P_{11}) \land (P_{21}) \land (P_{22}) \]

\[ := \chi_A^+ \]

\[ A \models \neg (P_{20}) := \chi_A^- \]
LSCP Walkthrough

Characterisation

\[ A = \langle \| r_7 \| = R_7, AP_6, N_5 \rangle \]

\[ A \models (P_{11}) \land (P_{21}) \land (P_{22}) \]
\[ := \chi_A^+ \]  

\[ A \not\models (P_{20}) \]
\[ := \chi_A^- \]
LSCP Walkthrough

Characterisation

\[ \mathcal{A} = \langle \| r_7 \| = R_7, AP_6, N_5 \rangle \]

\[ \mathcal{A} \models (P11) \land (P21) \land (P22) \]
\[ := \chi^+_{\mathcal{A}} \]

\[ \mathcal{A} \not\models (P20) \]
\[ := \chi^-_{\mathcal{A}} \]

\[ \mathcal{A} \models \chi^+_{\mathcal{A}} \land \chi^-_{\mathcal{A}} \]
LSCP Walkthrough

Projection

Algorithm

\[
\text{for (span=1 to num\_words)} \\
\quad \text{for (offset=1 to num\_words-span+1)} \\
\quad \quad \text{for (every assignment A in [offset..end])} \\
\quad \quad \quad X \leftarrow \text{characterisation}(A) \\
\quad \quad \quad C \leftarrow \text{projection}(X) \\
\quad \quad \text{for (every x in C)} \\
\quad \quad \quad \text{if (merit(x) >= pi[offset, span, C]) then} \\
\quad \quad \quad \quad \text{pi[offset, span, C] \leftarrow \{x, merit(x)\}}
\]

\[
A \models NP_{7}\text{.CAT = NP} \rightarrow \chi_A^+ \land \chi_A^-
\]
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)}

\[ \mathcal{A} \models NP_7.cat = NP \rightarrow \chi_\mathcal{A}^+ \land \chi_n^- \]
## LSCP Walkthrough

### Memoization

<table>
<thead>
<tr>
<th></th>
<th>Le juge octroie bref entretien à ce plaignant</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The judge grants brief interview to this plaintiff</td>
</tr>
<tr>
<td>2</td>
<td>*NP&lt;sub&gt;7&lt;/sub&gt; ( \rightarrow ) AP&lt;sub&gt;6&lt;/sub&gt; N&lt;sub&gt;5&lt;/sub&gt; A&lt;sub&gt;4&lt;/sub&gt;</td>
</tr>
</tbody>
</table>
LSCP
Solution Parse

```
S_{15}

NP_{3}

D_1

Le

The

| N_2

| juge

| judge

V_8

| octroie

| grants

| AP_{6}

| entretien

| interview

*NP_{7}

| A_{4}

| bref

| brief

VP_{9}

PP_{10}

P_{11}

| à

| to

| ce

| this

| N_{12}

N_{14}

NP_{12}

| plaignant

| plaintiff
```
Evaluation of *Numbat*

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Numbat</em></td>
<td>78.4%</td>
<td>70.6%</td>
<td>74.2%</td>
</tr>
</tbody>
</table>
### Evaluation of *Numbat*

**EASY**

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Numbat</em></td>
<td>78.4%</td>
<td>70.6%</td>
<td>74.2%</td>
</tr>
<tr>
<td>LPL Shallow parser</td>
<td>78.5%</td>
<td>83.8%</td>
<td>81%</td>
</tr>
<tr>
<td>LPL stochastic parser</td>
<td>90.1%</td>
<td>89.8%</td>
<td>89.9%</td>
</tr>
</tbody>
</table>
### Evaluation of *Numbat*

EASY

---

#### Example

<table>
<thead>
<tr>
<th>j’entendais</th>
<th>encore,</th>
<th>au fond</th>
<th>du pavillon,</th>
<th>monsieur</th>
<th>qui</th>
<th>essayait</th>
<th>d’ébranler</th>
<th>la porte.</th>
</tr>
</thead>
<tbody>
<tr>
<td>NV</td>
<td>GR</td>
<td>GP</td>
<td>GP</td>
<td>GN</td>
<td>GN</td>
<td>NV</td>
<td>PV</td>
<td>GN</td>
</tr>
<tr>
<td>VP</td>
<td>AdvP</td>
<td>PP</td>
<td>PP</td>
<td>NP</td>
<td>NP</td>
<td>VP</td>
<td>Prep. VP</td>
<td>NP</td>
</tr>
</tbody>
</table>

‘I was still hearing, at the back of the house, Sir, who was trying to shake the door.’
Evaluation of *Numbat*

Quasi-expressions

![Diagram of quasi-expressions](attachment://diagram.png)
Evaluation of *Numbat*

**Quasi-expressions**

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Numbat</em></td>
<td>74%</td>
<td>68%</td>
<td>71%</td>
</tr>
</tbody>
</table>

**Evaluation**

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Numbat</em></td>
<td>74%</td>
<td>68%</td>
<td>71%</td>
</tr>
</tbody>
</table>

**Knowledge Representation**

**Loose Satisfaction Chart Parsing**

**Conclusion**

Intersective Gradience and Optimality

Subsective Gradience and Acceptability

Outline

Background

Jean-Philippe Prost
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

1. Background
   - Gradience
   - Modelling Gradience
   - Classification

2. Intersective Gradience and Optimality
   - Knowledge Representation
   - Loose Satisfaction Chart Parsing
   - Evaluation

3. Subsective Gradience and Acceptability
   - Predicting Acceptability
   - Empirical Investigation
   - Interpretation

4. Conclusion
Problem

Subsective Gradience

Can we predict how grammatically acceptable an utterance is?
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
   - $N_c^+$
3 Constraint Weighting

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

4 Constructional Complexity
- \( T_C \): Total number of constraints specifying the construction \( C \)
- \( E_c \): 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 ($\gamma$)
- Taxed Cohesion ($\gamma'$)
### Human Judgements by Error Patterns

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Sentence</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>No violations</td>
<td>Marie a emprunté un très long chemin pour le retour</td>
<td>0.465</td>
</tr>
<tr>
<td>NP-violations</td>
<td>Marie a emprunté très long chemin un pour le retour</td>
<td>-0.643</td>
</tr>
<tr>
<td>AP-violations</td>
<td>Marie a emprunté un très long long chemin pour le retour</td>
<td>-0.216</td>
</tr>
<tr>
<td>PP-violations</td>
<td>Marie a emprunté un très long chemin le retour</td>
<td>-0.213</td>
</tr>
<tr>
<td>VP-violations</td>
<td>Marie emprunté un très long chemin pour le retour</td>
<td>-0.322</td>
</tr>
</tbody>
</table>
Model Fit

\[
\rho = 0.5381
\]
Model Fit

\[ \rho = 0.5425 \]
## Scale of $\gamma$-scores

<table>
<thead>
<tr>
<th>Type</th>
<th>2.3</th>
<th>1.1</th>
<th>5.2</th>
<th>3.2</th>
<th>4.2</th>
<th>3.1</th>
<th>2.2</th>
<th>4.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma$</td>
<td>4.11</td>
<td>3.96</td>
<td>3.93</td>
<td>3.91</td>
<td>3.78</td>
<td>3.64</td>
<td>3.60</td>
<td>3.33</td>
</tr>
</tbody>
</table>

| Ref. Rank | 15 | 1 | 5 | 7 | 3 | 10 | 4 | 6 |
| Rank | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

<table>
<thead>
<tr>
<th>Type</th>
<th>2.4</th>
<th>3.3</th>
<th>5.4</th>
<th>4.1</th>
<th>2.1</th>
<th>5.1</th>
<th>4.4</th>
<th>5.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma$</td>
<td>3.25</td>
<td>3.042</td>
<td>2.91</td>
<td>2.77</td>
<td>2.43</td>
<td>2.26</td>
<td>2.16</td>
<td>1.34</td>
</tr>
</tbody>
</table>

| Ref. Rank | 2 | 13 | 8 | 12 | 14 | 11 | 9 | 16 |
| Rank | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
## Scale of $\gamma$-scores

<table>
<thead>
<tr>
<th>Type</th>
<th>2.3</th>
<th>1.1</th>
<th>5.2</th>
<th>3.2</th>
<th>4.2</th>
<th>3.1</th>
<th>2.2</th>
<th>4.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma$</td>
<td>4.11</td>
<td>3.96</td>
<td>3.93</td>
<td>3.91</td>
<td>3.78</td>
<td>3.64</td>
<td>3.60</td>
<td>3.33</td>
</tr>
<tr>
<td>Ref. Rank</td>
<td>15</td>
<td>1</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>10</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Rank</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>2.4</th>
<th>3.3</th>
<th>5.4</th>
<th>4.1</th>
<th>2.1</th>
<th>5.1</th>
<th>4.4</th>
<th>5.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma$</td>
<td>3.25</td>
<td>3.042</td>
<td>2.91</td>
<td>2.77</td>
<td>2.43</td>
<td>2.26</td>
<td>2.16</td>
<td>1.34</td>
</tr>
<tr>
<td>Ref. Rank</td>
<td>2</td>
<td>13</td>
<td>8</td>
<td>12</td>
<td>14</td>
<td>11</td>
<td>9</td>
<td>16</td>
</tr>
<tr>
<td>Rank</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
</tbody>
</table>
## Scale of $\gamma$–scores

<table>
<thead>
<tr>
<th>Type</th>
<th>2.3</th>
<th>1.1</th>
<th>5.2</th>
<th>3.2</th>
<th>4.2</th>
<th>3.1</th>
<th>2.2</th>
<th>4.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma$</td>
<td>4.11</td>
<td>3.96</td>
<td>3.93</td>
<td>3.91</td>
<td>3.78</td>
<td>3.64</td>
<td>3.60</td>
<td>3.33</td>
</tr>
<tr>
<td>Ref. Rank</td>
<td>15</td>
<td>1</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>10</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Rank</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>2.4</th>
<th>3.3</th>
<th>5.4</th>
<th>4.1</th>
<th>2.1</th>
<th>5.1</th>
<th>4.4</th>
<th>5.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma$</td>
<td>3.25</td>
<td>3.042</td>
<td>2.91</td>
<td>2.77</td>
<td>2.43</td>
<td>2.26</td>
<td>2.16</td>
<td>1.34</td>
</tr>
<tr>
<td>Ref. Rank</td>
<td>2</td>
<td>13</td>
<td>8</td>
<td>12</td>
<td>14</td>
<td>11</td>
<td>9</td>
<td>16</td>
</tr>
<tr>
<td>Rank</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
</tbody>
</table>
Missing VP (5.3)
Missing NP (2.3)

S
   /\   
  NP VP
     |   |
  D  N  V
|  |  |
Les mègères ont
The shrews

V
  /\ |
  V NP PP
     |  |
  V envoyé une
 sent a

AP A PP
|  |  |
Adv A D NP
très grossière à
very rude/rough to

N N
|  |
leur voisin
their neighbour
## Ranking

<table>
<thead>
<tr>
<th>Rank diff.</th>
<th>Occurrences</th>
<th>% of total</th>
</tr>
</thead>
<tbody>
<tr>
<td>≥ 6</td>
<td>3</td>
<td>18.75%</td>
</tr>
<tr>
<td>= 4</td>
<td>1</td>
<td>6.25%</td>
</tr>
<tr>
<td>≤ 3</td>
<td>12</td>
<td>75%</td>
</tr>
</tbody>
</table>
Outline

1 Background
   - Gradience
   - Modelling Gradience
   - Classification

2 Intersective Gradience and Optimality
   - Knowledge Representation
   - Loose Satisfaction Chart Parsing
   - Evaluation

3 Subsective Gradience and Acceptability
   - Predicting Acceptability
   - Empirical Investigation
   - Interpretation

4 Conclusion
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