Minimalist Parsing: simplicity and feature unification

Sandiway Fong
Departments of Linguistics and Computer Science
University of Arizona
Tucson AZ USA

Conference on Language and Recursion. University of Mons, Belgium. March 14-16th 2011
Linguistic Theory

Computation and Linguistic Theory

• Principles and Parameters Approach (Chomsky 1980)
  – Linguistic principle = declarative statement:
    • Noun phrases must receive (abstract) Case (*Case Filter*)
    • An anaphor must be A-bound in its Governing Category (*Binding Principle A*)

• Minimalist Program (e.g. Chomsky 2001)
  – Precisely specified sequence of (feature-driven) operations
    • Merge: external and internal (= Move)
    • Agree: Probe and Goal
      (interaction between uninterpretable and interpretable features)
Principles and Parameters Approach

Theory is (relatively) unconstrained
- declarative principles can be “coded up” and combined in many different ways and yet remain “faithful” to the theory

PAPPI: Fong (1990)
Principles and Parameters Approach

Generate and test:
33 candidate parses examined by the parser

most are eliminated early by Case/Theta theory sub-system
Minimalist Program

Theory is highly constrained
• at each Merge step, operations are precisely specified, not much “wriggle room” for implementation

Prior work on grammar formalisms for MP theories e.g.
(Stabler, 1998)
(Lecomte & Retoré, 2001)
A Simple Implementation

• Take a theory in the Minimalist Program
  – e.g. Derivation by Phase (Chomsky 2001)

• What is the simplest possible computational implementation that we can get away with?

Simple does not necessarily equal efficient or “minimal”
Example

• Task: sorting a list of numbers \( n \): size of list
  – 5
  – 3
  – 8
  – 1
  – 7
  – 9
  – 4 ...

• Simplest implementation
  – only operation is to front lowest number
  – \( O(n^2) \) comparisons

• Quicksort
  – recursively sort around a \textit{pivot} number
  – same worst case but typically fast with \( O(n \lg n) \) comparisons

Scaling is important in sorting but for linguistic computation?
Example

• Phrase Structure Grammar
  – S $\rightarrow$ NP VP
  – VP $\rightarrow$ V NP
  – VP $\rightarrow$ V
  – VP $\rightarrow$ VP PP
  – NP $\rightarrow$ D N
  – NP $\rightarrow$ N
  – PP $\rightarrow$ P NP

• Use the grammar directly
  – recursive descent
  – bottom-up

• Transform the grammar into a finite state machine with a stack
  – Earley’s algorithm (as you go)
  – LR(k) Parsing (offline is for free)

Search space: Minimize the length of the derivation
Basic Computation

• start with lexical array of syntactic objects: \{\alpha, \ldots, \omega\}

• **Merge** "an indispensible operation of a recursive system"

• (external)
  – two syntactic objects (SOs): \alpha, \beta
  – create merged SO: \{\alpha, \beta\}
  – \text{label}\{\alpha, \beta\} = \text{label}(\alpha) \text{ or } \text{label}(\beta)

• (internal), implements **Displacement**
  – SOs: \alpha and \gamma, \gamma properly contained in \alpha
  – create SO: \{\alpha, \gamma\}
  – \text{label}\{\alpha, \gamma\} = \text{label}(\alpha)

• **Agree**
  – active probe SO: \alpha (active = uninterpretable features), goal SO: \beta
  – match and delete uninterpretable features of probe and goal

• **Convergent derivation:** uninterpretable features must be eliminated
Basic Implementation (1)

Definite clause grammar (DCG) (simplified)

\[ V([V V N]) \rightarrow V(V), n(N). \]
\[ V([V V \text{Verb}]) \rightarrow [\text{Verb}]. \]

- phonetic matrix: \( f(\text{pmatrix}, \text{like}) \)
- (takes an) argument: \( f(\text{arg}, +) \)
- uninterpretable Case: \( f(\text{case}, _) \)

% (big V) verb classes
\[ bV(n('V', [], [V, N])) \rightarrow bV0(V), n0(N), \{ \text{theta}(V), \text{theta}(N) \} \] report 'theta merge V & N'.
\[ bV0(n('V', [f(\text{pmatrix}, \text{Verb}), f(\text{arg}, +)], [])) \rightarrow [\text{Verb}], \{ \text{transitive}(\text{Verb}) \}. \]
\[ bV0(n('V', [f(\text{pmatrix}, \text{Verb}), f(\text{arg}, +)], [])) \rightarrow [\text{Verb}], \{ \text{unaccusative}(\text{Verb}) \}. \]
\[ bV0(n('V', [f(\text{pmatrix}, \text{Verb})], [])) \rightarrow [\text{Verb}], \{ \text{unergative}(\text{Verb}) \}. \]

transitive(\textit{like}). transitive(\textit{expect}).
unergative(\textit{run}). unaccusative(\textit{arrive}).

\[ n0(n(n, [f(\text{pmatrix}, \text{BNoun}) | \text{Fs}], [])) \rightarrow [\text{BNoun}], \{ \text{bareNoun}(\text{BNoun}, \text{Fs}) \}. \]
bareNoun(\textit{john}, [f(\text{phi}, 3-sg-m), f(\text{case}, _), f(\text{arg}, +)]).
Basic Implementation (2)

Definite clause grammar (DCG) (*simplified*)

\[ v([v \ N \ v]) \rightarrow n(N), \ v(v). \]
\[ v([v \ v \ V]) \rightarrow v(v), \ V(V). \]
\[ v([v^*]) \rightarrow \[]. \]

– features (\(v^*\))
  
  • uninterpretable \(\phi\)-features: \(f(\phi,\_\_\_\_)\)
  
  • can value accusative Case: \(f(\text{case},\text{acc})\)

\%
\(v^*/vP\)
\(v(n(v, [], [N, V])) \rightarrow n0(N), \ \{\text{theta}(N)\}, \ v1(V) \) report 'theta merge N & v'.
\(v1(n(v, [], [V, BV])) \rightarrow v0(V), \ bV(BV), \ \{\text{goals}(BV, Goals)\}, \ \text{agreed}(V, Goals)\) report 'merge v & V'.
\(v0(n('v*', [f(\phi,\_\_\_), f(\text{case},\text{acc})], [])) \rightarrow \[].\)
Basic Implementation (2)

- \text{Agree}(v^*,N)

\textbf{Operation: unification} (Robinson, 1965) (match and instantiate unvalued features)

<table>
<thead>
<tr>
<th>Probe</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{f}(\phi,_____)</td>
<td>\text{f}(\phi,3\text{-sg-f})</td>
</tr>
<tr>
<td>\text{f}(\text{case},\text{acc})</td>
<td>\text{f}(\text{case},__)</td>
</tr>
</tbody>
</table>

Uninterpretable/unvalued features (\textit{represented by variables}) are eliminated
Basic Implementation (3)

Definite clause grammar (DCG)

\[
\begin{align*}
T([T N v]) & \rightarrow n(N), v(v). \\
T([T N v]) & \rightarrow v(v), \{N \text{ a goal}\} \\
T([T]) & \rightarrow [].
\end{align*}
\]

– features (φ-complete T)

- uninterpretable φ-features: \(f(\phi,\_\_\_)\)
- can value nominative Case: \(f(\text{case},\text{nom})\)
- EPP

---

\[
\begin{align*}
t(n(t,\[],[N,T])) & \rightarrow n0(N), \{\text{nonarg}(N)\}, t1(T,\_\_) \text{ report 'merge expl & T'.} \\
\% EPP: (1) merge \\
t(n(t,\[],[\text{Goal},T])) & \rightarrow t1(T,\text{Goal}) \text{ report 'move to spec-T'.} \\
\% EPP: (2) move with maximize matching
\end{align*}
\]

\[
\begin{align*}
t1(n(t,\[],[T,V]),G) & \rightarrow t0(T), v(V), \{\text{goals}(V,\text{Goals}), \text{agree}(T,\text{Goals})\}, \text{Goals} = [\text{G}_1\_\_] \text{ report 'merge T & v'.} \\
t0(n(t, [f(\phi,\_\_\_), f(\text{case},\text{nom})],[])) & \rightarrow [].
\end{align*}
\]
Putting it all together (1)

1. theta merge V & N
2. merge v & V
3. theta merge N & v
4. merge T & v
5. move to spec-T
6. merge C & T

V

V

like

n

mary
Putting it all together (2)

1. theta merge V & N
2. merge v & V
3. theta merge N & v
4. merge T & v
5. move to spec-T
6. merge C & T

Probe [v*] agrees with goal [n mary]
Putting it all together (3)
Putting it all together (4)

1. theta merge V & N
2. merge v & V
3. theta merge N & v
4. merge T & v
5. move to spec-T
6. merge C & T

Probe [t] agrees with goal [n john]
Putting it all together (5)

1. theta merge V & N
2. merge v & V
3. theta merge N & v
4. merge T & v
5. move to spec-T
6. merge C & T

(EPP requirement)
Putting it all together (6)

Operation: Spell-Out
(not currently implemented)

“John likes Mary”
only one copy of John is pronounced
bundle

$$T + v^* f(\text{phi}, 3\text{-sg-f}) + v(\text{like}) = \text{likes}$$
Examples

assume, are raising constructions and their exceptional-Case-marking (ECM) counterparts, as shown schematically in (4a), where \( \beta \) is the matrix clause, \( \alpha \) is an infinitival with YP a verbal phrase (the case most relevant here), and P is the probe: T with a raising verb (case (4b)), \( \nu \) with an ECM transitive verb (case (4c)).

\[(4)\]
\[
\text{a. } [\beta \text{ P } [\alpha \text{ [Subj [H YP]]}] ]
\]
\[
\text{b. i. there are likely to be awarded several prizes}
\]
\[
\text{ii. several prizes are likely to be awarded}
\]
\[
\text{c. i. we expect there to be awarded several prizes}
\]
\[
\text{ii. we expect several prizes to be awarded}
\]

The Case/agreement properties of Subj in (4a), and its overt location, are determined by properties of the matrix probe P, not internally to \( \alpha \). \( \alpha \) is a TP with defective head \( T_{\text{def}} \), which is unable to determine Case/agreement but has an EPP-feature, overtly manifested in (4c). Raising-ECM parallels give good reason to believe that the EPP-feature is manifested in (4b) as well, by trace of the matrix subject; preference for Merge over (more complex) Move gives a plausible reason for the surface distinction between [Spec, \( T_{\text{def}} \)] in (4b) and in (4c) (see MI). In (4bi) and (4ci), the EPP-feature of \( T_{\text{def}} \) is satisfied by Merge of expletive; in (4bi) and (4ci), by raising of the direct object.

(Chomsky 2001)
A Worked Example

Consider the derivation of

- several prizes are likely to be awarded \((= 4(b)(ii))\)
  
  \[\textit{awarded} = \textit{award} + \text{-ed} \text{ (adjectival participle)}\]

- \text{-ed} \ \phi\text{–incomplete: uninterpretable Number and Gender only}
  
  uninterpretable Case

  morphologically unrealized in English (cf. Icelandic)

?- parse([\text{be,likely,be,ed,award,several,prizes}]).

Probe [a!case ed] agrees with goal [n!case several prizes]  
Probe [tdef] agrees with goal [n!case several prizes]  
Probe [t] agrees with goal [n several prizes]  
[c[c][t[n several prizes][t[t][v[v be][a[a likely][t[n several prizes][t[tdef][v[v be][a[a ed][V[V award][n several prizes]]]]]]]]]]]

\text{Agree(a,N)}

\text{-ed: } \phi, \text{ Case}

\text{N: } \phi, \text{ Case}
A Worked Example

V
/    \
|     |
award several prizes
A Worked Example

Agree(a,N)
-\textit{ed}: \Phi, \textbf{Case}
N: \Phi, \textbf{Case}

Notation: !case means feature is unvalued
A Worked Example

Agree(Tdef,N)
Tdef: φ
N: φ, Case
A Worked Example

(several prizes raises to subject position of embedded infinitival)
A Worked Example

Agree(T,N)
T: φ, Nominative
N: φ, Case

Case for –ed also valued because of earlier unification step

Agree(a,N)
-ed: φ, Case
N: φ, Case

Unification presents an possible advantage: computation is more local
Probing with multiple goals

What about Prt? Its φ-features are deleted at stage α and should therefore be invisible to Match by the probe. Case of Prt cannot be valued and the derivation crashes, contrary to fact.

The problem is overcome if Spell-Out takes place at the strong-phase level. Then the φ-features of Prt are still visible at stage β of the cycle, though deleted; they disappear at the strong-phase level CP or vP, as the phase is transmitted to the phonological component. At stage α of the cycle, the φ-features of Prt are valued by Prt-DO matching, as just discussed. At the next stage, the probe T/ν matches the (still visible) goal Prt, valuing its Case feature; and the probe matches the goal DO, valuing the Case feature of DO as well as its own features (since DO is φ-complete).

compare with...

(Chomsky 2001)
A Worked Example

Spell-Out
Several prizes are likely to be awarded
Another Example

Consider also the derivation of

- There are likely to be awarded several prizes   (= 4(b)(i))

?- parse([be,likely,there,be,ed,award,several,prizes]).
Probe [a!case ed] agrees with goal [n!case several prizes]
Probe [t!def] agrees with goal [n!case several prizes]
Probe [t!phi] agrees with goal [n!phi there]
Probe [t] agrees with goal [n several prizes]
"c[c][t[n there][t[t][v[v be][a[a likely][t[n there][t[tdef][v[v be][a[a ed][V[V award][n several prizes]]]]]]]]]]]"

pleonastic there: φ-incomplete (Person only)

Agree(a,N) -ed: φ, Case
N:   φ, Case

Agree(T,N)
T:   φ
N:   φ
Another Example

Agree(a,N)
-ed: φ, Case
N: φ, Case
Another Example

$\text{1}^{st}$ Agree(T,N)  
T: $\phi$, Nominative  
N: $\phi$

$\text{2}^{nd}$ Agree(T,N)  
T: $\phi$, Nominative  
N: $\phi$, Case

*there* : $\phi$-feature indirectly valued by $\text{2}^{nd}$ Agree(T,N)  
*-ed* : Case also indirectly valued by $\text{2}^{nd}$ Agree(T,N)  
However, T must probe past *there* all the way on down
Another Example

Spell-Out
There are likely to be awarded several prizes

(there raises to matrix subject position)
Examples

assume, are raising constructions and their exceptional-Case-marking (ECM) counterparts, as shown schematically in (4a), where $\beta$ is the matrix clause, $\alpha$ is an infinitival with YP a verbal phrase (the case most relevant here), and P is the probe: T with a raising verb (case (4b)), $\nu$ with an ECM transitive verb (case (4c)).

(4) a. $[\beta \text{ P } [\alpha \text{ Subj } [\text{ H YP }]]]$
   b. i. there are likely to be awarded several prizes
      ii. several prizes are likely to be awarded
   c. i. we expect there to be awarded several prizes
      ii. we expect several prizes to be awarded

[Spec, $T_{\text{def}}$] in (4b) and in (4c) (see MI). In (4bi) and (4ci), the EPP-feature of $T_{\text{def}}$ is satisfied by Merge of expletive; in (4bii) and (4cii), by raising of the direct object.

(Chomsky 2001)
Example: *we expect several prizes to be awarded*
Example: *we expect several prizes to be awarded*
Conclusions

• described a (non-trivial) implementation of a probe-goal system for Case and verbal inflection

• we want to (1) minimize # operations, and (2) localize goal search as far as possible

• unification for uninterpretable feature valuation can help both (1) and (2)