Modeling Linguistic Theory on a Computer: From GB to Minimalism

Sandiway Fong
Dept. of Linguistics
Dept. of Computer Science
Outline

• Mature system: PAPPI
  – parser in the principles-and-parameters framework
  – principles are formalized and declaratively stated in Prolog (logic)
  – principles are mapped onto general computational mechanisms
  – recovers all possible parses
  – (free software, recently ported to MacOS X and Linux)
  – (see http://dingo.sbs.arizona.edu/~sandiw/) 

• Current work
  – introduce a left-to-right parser based on the probe-goal model from the Minimalist Program (MP)
  – take a look at modeling some data from SOV languages
    • relativization in Turkish and Japanese
    • psycholinguistics (parsing preferences)
  – (software yet to be released...)
PAPPI: Overview

- user’s viewpoint

parser operations corresponding to linguistic principles (= theory)

sentence

syntactic representations
PAPPI: Overview

- **parser operations can be**
  - turned on or off
  - metered
- **syntactic representations can be**
  - displayed
  - examined
  - in the context of a parser operation
  - dissected
  - features displayed
PAPPI: Coverage

• **supplied with a basic set of principles**
  - X’-based phrase structure, Case, Binding, ECP, Theta, head movement, phrasal movement, LF movement, QR, operator-variable, WCO
  - handles a couple hundred English examples from Lasnik and Uriagereka’s (1988) *A Course in GB Syntax*

• **more modules and principles can be added or borrowed**
  - VP-internal subjects, NPIs, double objects  *Zero Syntax* (Pesetsky, 1995)
  - Japanese (some Korean): head-final, pro-drop, scrambling
  - Dutch (some German): V2, verb raising
  - French (some Spanish): verb movement, pronominal clitics
  - Turkish, Hungarian: complex morphology
  - Arabic: VSO, SVO word orders
PAPPI: Architecture

- software layers
PAPPI: Architecture

- software layers

- competing parses can be run in parallel across multiple machines
PAPPI: Machinery

- **morphology**
  - simple morpheme concatenation
  - morphemes may project or be rendered as features

(Example from the Hungarian implementation)
PAPPI: LR Machinery

- **specification**
  - rule $XP \rightarrow [XB|\text{spec}(XB)]$ ordered $\text{specFinal}(XP)$, $\text{proj}(XB,XP)$.
  - rule $XB \rightarrow [X|\text{compl}(X)]$ ordered $\text{headInitial}(X)$ st $\text{bar}(XB)$, $\text{proj}(X, XB)$, $\text{head}(X)$.
  - rule $v(V)$ moves_to i provided $\text{agr}$(strong), $\text{finite}(V)$.
  - rule $v(V)$ moves_to i provided $\text{agr}$(weak), $V$ has _feature aux.

- **implementation**
  - bottom-up, shift-reduce parser
  - push-down automaton (PDA)
  - stack-based merge
    - shift
    - reduce
  - canonical LR(1)
    - disambiguate through one word lookahead

- **phrase structure**
  - parameterized $X'$-rules
  - head movement rules

rules are not used directly during parsing for computational efficiency
- mapped at compile-time onto LR machinery

S \rightarrow . NP VP
NP \rightarrow . D N
NP \rightarrow . N
NP \rightarrow . NP PP

State 0

NP \rightarrow D . N

State 1

S \rightarrow NP . VP
NP \rightarrow NP . PP
NP \rightarrow . D N
NP \rightarrow . N
NP \rightarrow . NP PP

State 2

NP \rightarrow N .

State 3

S \rightarrow NP . VP
NP \rightarrow NP . PP
NP \rightarrow . D N
NP \rightarrow . N
NP \rightarrow . NP PP
VP \rightarrow . V NP
VP \rightarrow . V
VP \rightarrow . VP PP
PP \rightarrow . P NP

State 4

State 1

NP \rightarrow D . N

State 2

NP \rightarrow N .

State 3

S \rightarrow NP . VP
NP \rightarrow NP . PP
NP \rightarrow . D N
NP \rightarrow . N
NP \rightarrow . NP PP

State 4
PAPPI: Machine Parameters

- **specification**
  - coindexSubjAndINFL in_all_configurations CF where specIP(CF,Subject) then coindexSI(Subject,CF).
  - subjacency in_all_configurations CF where isTrace(CF), upPath(CF,Path) then lessThan2BoundingNodes(Path)

- **implementation**
  - use type inferencing defined over category labels
    - figure out which LR reduce actions should place an outcall to a parser operation
  - subjacency can be called during chain aggregation

- selected parser operations may be integrated with phrase structure recovery or chain formation
  - machine parameter
  - however, not always efficient to do so
PAPPI: Chain Formation

• specification
  – assignment of a chain feature to constituents

chain(NP[1], last, [[pp, vp, vp, i1], []])
chain(NP[1], medial, [[s2, vp, np, i1, i2, s1], []])
chain([], head, [])

• combinatorics
  – exponential growth

<table>
<thead>
<tr>
<th>NPs</th>
<th>Indexings</th>
<th>NPs</th>
<th>Indexings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>7</td>
<td>877</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>8</td>
<td>4140</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>9</td>
<td>21147</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>10</td>
<td>115575</td>
</tr>
<tr>
<td>5</td>
<td>52</td>
<td>11</td>
<td>678570</td>
</tr>
<tr>
<td>6</td>
<td>263</td>
<td>12</td>
<td>4123597</td>
</tr>
</tbody>
</table>

Upper-bounded by Bell’s Exponential Number

\[ B_n = \sum_{m=1}^{n} \sum_{k=0}^{m} \frac{(-1)^{m-k}}{(m-k)! k!} k^n \]

\[ m_n e^{m_n - n - \frac{1}{2}} \sqrt{\ln n} \]

\[ m_n \ln m_n = n - \frac{1}{2} \]

• recovery of chains
  – compute all possible combinations
  – each empty category optionally participates in a chain
  – each overt constituent optionally heads a chain
PAPPI: Chain Formation

• specification
  – assignment of a chain feature to constituents
    \[
    \begin{align*}
    \text{chain}(\text{NP}[1], \text{last}, [[\text{pp}, \text{vp}, \text{vp}, \text{i1}], []]) \\
    \text{chain}(\text{NP}[1], \text{medial}, [[\text{i2}, \text{vp}, \text{vp}, \text{i1}, \text{i2}, \text{c1}], []]) \\
    \text{chain}([], \text{head}, [])
    \end{align*}
    \]

• implementation
  – possible chains compositionally defined
  – incrementally computed
  – bottom-up

  – allows parser operation merge

• recovery of chains
  – compute all possible combinations
    • each empty category optionally participates in a chain
    • each overt constituent optionally heads a chain

\[
\begin{align*}
\text{CF} & = \left( C_X \right) \odot \left( C_Y \right) \\
\text{Compositional Definition} & \text{ “optional cross-product”}
\end{align*}
\]
PAPPI: Chain Formation

- **specification**
  - assignment of a chain feature to constituents

```plaintext
chain(NP[1], last, [[pp, vp, vp, i1], []])
chain(NP[1], medial, [[c2, vp, vp, i1, i2, c1], []])
chain([], head, [])
```

- **merge constraints on chain paths**

```plaintext
m Subjacency
m Lowering Filter
i Wh-movement in Syntax
```

- loweringFilter in_all_configurations CF where isTrace(CF), downPath(CF, Path) then Path=[].
- subjacency in_all_configurations CF where isTrace(CF), upPath(CF, Path) then lessThan2BoundingNodes(Path)

- **recovery of chains**
  - compute all possible combinations
    - each empty category optionally participates in a chain
    - each overt constituent optionally heads a chain
PAPPI: Domain Computation

• specification
  – gc(X) smallest_configuration CF st cat(CF,C), member(C,[np,i2])
  – with_components
  – X,
  – G given_by governs(G,X,CF),
  – S given_by accSubj(S,X,CF).

• implementing
  – Governing Category (GC):
  – GC(α) is the smallest NP or IP containing:
  – (A) α, and
  – (B) a governor of α, and
  – (C) an accessible SUBJECT for α.

• minimal domain
  – incremental
  – bottom-up
PAPPI: Domain Computation

• **specification**
  - gc(X) *smallest_configuration* CF st cat(CF,C), member(C,[np,i2])
  - **with_components**
  - X,
  - G *given_by* governs(G,X,CF),
  - S *given_by* accSubj(S,X,CF).

• **minimal domain**
  - incremental
  - bottom-up

• **used in**
  - Binding Condition A
    • An anaphor must be A-bound in its GC
  - conditionA *in_all_configurations* CF where
    - anaphor(CF) then gc(CF,GC), aBound(CF,GC).
    - anaphor(NP) :- NP has_feature apos, NP has_feature a(+).
Probe-Goal Parser: Overview

- **strictly incremental**
  - left-to-right
  - uses elementary tree (eT) composition
    - guided by selection
    - open positions filled from input
  - epp
  - no bottom-up merge/move

- **probe-goal agreement**
  - uninterpretable interpretable feature system
Probe-Goal Parser: Selection

- **recipe**
  - start(c)
  - pick eT headed by c from input (or M)
    - fill Spec, run agree(P,M)
    - fill Head, update P
    - fill Comp (c select c’, recurse)

- **example**

- **select drives derivation**
  - left-to-right

- **memory elements**
  - MoveBox (M)
    - emptied in accordance with theta theory
  - filled from input
  - ProbeBox (P)
    - current probe
Probe-Goal Parser: Selection

- **recipe**
  start(c)
  pick eT headed by c from input (or M)
  - fill Spec, run agree(P,M)
  - fill Head, update P
  - fill Comp (c select c’, recurse)

- **example**

- **select drives**
  derivation
  - left-to-right

- **memory elements**
  - MoveBox (M)
    - emptied in accordance with theta theory
    - filled from input
  - ProbeBox (P)
    - current probe

- **note**
  - extends derivation to the right
  - similar to Phillips (1995)
Probe-Goal Parser: Selection

- **recipe**
  - start(c)
  - pick eT headed by c
  - from input (or M)
    - fill Spec, run agree(P,M)
    - fill Head, update P
    - fill Comp (c select c’, recurse)

- **example**

- **select drives derivation**
  - left-to-right

- **memory elements**
  - MoveBox (M)
    - emptied in accordance with theta theory
    - filled from input
  - ProbeBox (P)
    - current probe

- **note**
  - no merge/move

**agree**
- $\phi$-features $\rightarrow$ probe
- case $\rightarrow$ goal
## Probe-Goal Parser: Lexicon

<table>
<thead>
<tr>
<th>lexical item</th>
<th>properties</th>
<th>uninterpretable features</th>
<th>interpretable features</th>
</tr>
</thead>
<tbody>
<tr>
<td>v* (transitive)</td>
<td>select(V)</td>
<td>per(P) (epp)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>spec(select(N))</td>
<td>num(N)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>value(case(acc))</td>
<td>gen(G)</td>
<td></td>
</tr>
<tr>
<td>v (unaccusative)</td>
<td>select(V)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>v# (unergative)</td>
<td>select(V)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>spec(select(N))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRT. (participle)</td>
<td>select(V)</td>
<td>num(N) case(C)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>gen(G)</td>
<td></td>
</tr>
<tr>
<td>V (trans/unacc)</td>
<td>select(N)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V (unergative)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V (raising/ecm)</td>
<td>select(T(def))</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Probe-Goal Parser: Lexicon

<table>
<thead>
<tr>
<th>lexical item</th>
<th>properties</th>
<th>uninterpretable features</th>
<th>interpretable features</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>select(v)</td>
<td>per(P) epp</td>
<td></td>
</tr>
<tr>
<td></td>
<td>value(case(nom))</td>
<td>num(N)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>gen(G)</td>
<td></td>
</tr>
<tr>
<td>T(def) (ϕ-incomplete)</td>
<td>select(v)</td>
<td>per(P) epp</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>select(T)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c(wh)</td>
<td>select(T)</td>
<td>q epp</td>
<td>wh</td>
</tr>
<tr>
<td>N (referential)</td>
<td>select(N)</td>
<td>case(C)</td>
<td></td>
</tr>
<tr>
<td>N (wh)</td>
<td></td>
<td>case(C) wh</td>
<td>per(P) q num(N) gen(G)</td>
</tr>
<tr>
<td>N (expl)</td>
<td>select(T(def))</td>
<td>per(P)</td>
<td></td>
</tr>
</tbody>
</table>
Probe-Goal Parser: Memory

- **MoveBox M Management Rules**
  - (implements theta theory)
  1. Initial condition: *empty*
  2. Fill condition: *copy from input*
  3. Use condition: *prefer M over input*
  4. Empty condition: *M emptied when used at selected positions. EXPL emptied optionally at non-selected positions.*

- **examples**
  from Derivation by Phase. Chomsky (1999)
  1. several prizes are likely to be awarded
     - [c [c] [T several prizes [T [T past(-)] [v [v be] [a [a likely] [T c(prizes) [T [T] [v [v PRT] [V [V award] c(prizes)])]]]]]]
  2. there are likely to be awarded several prizes
     - [c [c] [T there [T [T past(-)] [v [v be] [a [a likely] [T c(there) [T [T] [v [v prt] [V [V award] several prizes)])]]]]]
Probe-Goal Parser vs. PAPPI

- instrument parser operations
- examples
  1. several prizes are likely to be awarded
  2. there are likely to be awarded several prizes
# Probe-Goal Parser vs. PAPPI

## Instrument Parser Operations

<table>
<thead>
<tr>
<th>example</th>
<th>structure building</th>
<th>agree/move vs. move-α</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>15 eT/10 words</td>
<td>5/2</td>
</tr>
<tr>
<td>1. PAPPI</td>
<td>1864 LR ≈ 373 eT</td>
<td>26</td>
</tr>
<tr>
<td>2.</td>
<td>20 eT/16 words</td>
<td>7/7</td>
</tr>
<tr>
<td>2. PAPPI</td>
<td>1432 LR ≈ 286 eT</td>
<td>67</td>
</tr>
</tbody>
</table>

## Examples

1. Several prizes are likely to be awarded.
2. There are likely to be awarded several prizes.
Probe-Goal Parser: efficiency and preferences

- **MoveBox M Management Rule**
  3. Use condition: *prefer* M over input

- **efficiency**
  - choice point management
  - eliminate choice points

- *How to expand the left-to-right model to deal with SOV languages and parsing preferences?*
  - look at some relativization data from Turkish and Japanese
Probe-Goal Parser: SOV

- **assumptions**
  - posit simplex sentence structure
  - initially selection-driven
  - fill in open positions on left edge
    - left to right
  - possible continuations
    - 1: S O V  
      *simplex sentence*
    - 2: [ S O V ]-REL  
      *V complement clause*
    - 3: [ S O V ]  \(\Rightarrow\)  
      *N prenominal relative clause*

**note**
- don’t posit unnecessary structure
- relative clauses are initially processed as main clauses with dropped arguments
- 1 < 2 < 3, e.g. 2 < 3 for Japanese (Miyamoto 2002) (Yamashita 1995)
Probe-Goal Parser: SOV

- **assumptions**
  - posit simplex sentence structure
  - initially selection-driven
  - fill in open positions on left edge
    - left to right
  - possible continuations
    - 1: S O V \textit{simplex sentence}
    - 2: [ S O V ]-REL \textit{V complement clause}
    - 3: [ S O V ] \Rightarrow N \textit{prenominal relative clause}

**note**

- lack of expectation
  - \textit{in addition to the top-down (predictive) component}
  - needs to be a bottom-up component to the parser as well
Probe-Goal Parser: relative clauses

- **prenominal relative clause structure**
  - Turkish
    - [ S-GEN O V-OREL-AGR ] H
    - [ S O-ACC V-SREL ] H
    - OREL = -dUk
    - SREL = -An
  - Japanese
    - [ S-NOM O V ] H
    - [ S O-ACC V ] H
    - *no overt relativizer*

- **relativization preferences**
  - Turkish
    - *ambiguous Bare NP (BNP)*
    - BNP: BNP is object
    - BNP with possessive AGR: BNP is subject
  - Japanese
    - subject relative clauses easier to process
    - *scrambled object preference for relativization out of possessive object*
Ambiguity in Relativization (Turkish)

bare NPs and SREL

• schema
  – BNP V-SREL H

• notes
  – BNP = bare NP (not marked with ACC, same as NOM)
    • (1) indefinite object NP, i.e. \([O \ [ e \ BNP \ V-SREL ]] \Rightarrow H\)
    • (2) subject NP, i.e. \([O \ [ BNP \ e \ V-SREL ]] \Rightarrow H\)

general preference (subject relativization)
  – e BNP V-SREL H

• however ...
  – Object relativization preferred, i.e. BNP e V-SREL H when BNP V together form a unit concept, as in:
    • bee sting, lightning strike (pseudo agent incorporation)
Ambiguity in Relativization (Turkish)

possessor relativization and bare NPs

• schema
  – BNP-AGR V-SREL H (AGR indicates possessive agreement)

• example (Iskender, p.c.)
  – daughter-AGR see-SREL man
    the man whose daughter saw s.t./s.o.

general preference *(BNP as subject)*
  – [e BNP]-AGR pro V-SREL H

• notes
  – BNP with AGR in subject position vs. in object position without
  – Object pro normally disfavored viz-a-viz subject pro
  – See also (Güngördü & Engdahl, 1998) for a HPSG account
Possessor Relativization (Japanese)

subject/object asymmetry

- examples (Hirose, p.c.)
- also Korean (K. Shin; S. Kang, p.c.)
  - subject
    - musume-ga watashi-o mita otoko
    - [e daughter]-NOM I-ACC see-PAST man
    - the man whose daughter saw me
  - object
    - musume-o watashi-ga mita otoko
    - [e daughter]-ACC I-NOM e see-PAST man
    - ?I-NOM [e daughter]-ACC see-PAST man

- summary
  - scrambled version preferred for object relativization case
  - *non-scrambled version is more marked*
  - in object scrambling, object raises to spec-T (Miyagawa, 2004)
  - possible difference wrt. inalienable/alienable possession in Korean
Probe-Goal Parser: A Model

- **initial expectation**
  - simple clause
  - top-down prediction
  - fill in left edge
  - insert *pro* as necessary

- **surprise**
  - triggers REL insertion at head noun and bottom-up structure
  - REL in Japanese (covert), Turkish (overt)
  - S O V (REL) H

- **functions of REL**
  - introduces empty operator
  - looks for associated gap (find-e) in predicted structure

doesn’t work for Chinese:
object relativization preference (Hsiao & Gibson)