Computing Dynamic Meanings
Day 2: Introduction to syntactic and semantic parsing in ACT-R/pyactr

Jakub Dotlačil & Adrian Brasoveanu

ESSLLI 2018, August 7 2018
Topic for today

• Parsing

• Is interpretation incremental?
  Altmann and Steedman, 1988; Marslen-Wilson, 1973, 1975; Tanenhaus et al., 1995; Trueswell et al., 1994

• If so, what does it tell us about (syntactic) parsing?
  Steedman, Stabler, Shieber & Johnson
Syntactic parsers are incremental

Garden-path or “garden-path-like” effects:

(1) a. The horse raced past the barn fell.
    b. The horse that was raced past the barn fell.

(2)

(3)

(4)

(5) a. Put the apple on the towel into the box.
    b. Put the apple that is on the towel into the box.
Syntactic parsers are incremental

Garden-path or “garden-path-like” effects:

(1) a. The horse raced past the barn fell.
b. The horse that was raced past the barn fell.

(2) Trump urged to stop tweeting on Trump tower meeting.
Syntactic parsers are incremental.

Garden-path or “garden-path-like” effects:

(1)  a. The horse raced past the barn fell.
     b. The horse that was raced past the barn fell.

(2)  Trump urged to stop tweeting on Trump tower meeting.

(3)  a. While she mended a sock fell on the floor.
     b. While she mended, a sock fell on the floor.
Syntactic parsers are incremental

Garden-path or “garden-path-like” effects:

(1) a. The horse raced past the barn fell.
   b. The horse that was raced past the barn fell.

(2) Trump urged to stop tweeting on Trump tower meeting.

(3) a. While she mended a sock fell on the floor.
    b. While she mended, a sock fell on the floor.

(4) a. The professor saw the students walked across the quad.
    b. The professor saw that the students walked across the quad.
Syntactic parsers are incremental

Garden-path or “garden-path-like” effects:

(1)  a. The horse raced past the barn fell.
     b. The horse that was raced past the barn fell.

(2)  Trump urged to stop tweeting on Trump tower meeting.

(3)  a. While she mended a sock fell on the floor.
     b. While she mended, a sock fell on the floor.

(4)  a. The professor saw the students walked across the quad.
     b. The professor saw that the students walked across the quad.

(5)  a. Put the apple on the towel into the box.
     b. Put the apple that is on the towel into the box.
Syntactic parsers are incremental

Garden-path or “garden-path-like” effects:

I. The horse raced past the barn fell.
II. While she mended a sock fell on the floor.
III. The professor saw the students walked across the quad.
IV. Put the apple on the towel into the box.

→ syntactic parsing is incremental

Terminology

- locally/temporarily ambiguous sentences
- garden-path sentences
Marslen-Wilson (1975): shadowing

Normal sentence:

- The new peace terms have been announced. They call for the unconditional withdrawal of all the enemy forces.

Semantically anomalous sentence:

- The new peace terms have been announced. They call for the unconditional *universe* of all the enemy forces.
The word *withdrawal/universe*: disrupted in one of the syllables, or not disrupted

- 2nd syllable: *withdrawal* → *withdewal*
- 3rd syllable: *withdrawal* → *withdrawack*
- 2nd syllable: *universe* → *unopverse*
- 3rd syllable: *universe* → *unitierse*
Marslen-Wilson (1975): shadowing

- 2nd syllable: withdrawal → withdewal
- 3rd syllable: withdrawal → withdrawack

- Corrected often (27 cases)

- 2nd syllable: universe → unopverse
- 3rd syllable: universe → unitierse

- Hardly ever corrected (5 cases)
Marslen-Wilson (1975): shadowing

Item

_The new peace terms have been announced. They call for the unconditional withdrawal/universe of all the enemy forces._

- Incremental interpretation, hence, restoration of disrupted words only in normal condition (in 2nd and 3rd syllables)

- Problems?
Eye-tracking & 80’s ad 90’s

I had been invalided home from the Front

gaze duration/first pass

total time

re-reading time
Reduced-relative garden path

(6) The horse raced past the barn fell.
(7) The horse raced...

Main-clause $\gg$ reduced-relative clause

Minimal Attachment:
listeners/readers posit the smallest syntactic structures compatible with the input

Frazier, 1978

Closely-similar alternatives: J. Hale, 2011; Pritchett, 1988, 1992
Other possibilities: Bever, 1970; Gibson, 1991; J. Hale, 2001; Steedman, 2001
Interaction of semantic selectional restrictions and parsing

(8)  

a. The defendant examined by the lawyer turned out to be unreliable. (Animate Reduced)  
b. The defendant that was examined by the lawyer turned out to be unreliable. (Animate Unreduced)
Interaction of semantic selectional restrictions and parsing

(8) a. The defendant examined by the lawyer turned out to be unreliable. (Animate Reduced)
   b. The defendant that was examined by the lawyer turned out to be unreliable. (Animate Unreduced)

(9) a. The evidence examined by the lawyer turned out to be unreliable. (Inanimate Reduced)
   b. The evidence that was examined by the lawyer turned out to be unreliable. (Inanimate Unreduced)
I. The Animate/Inanimate $[V \text{ examined }] [\text{disamb by the lawyer}]$

II. The Animate/Inanimate that was $[V \text{ examined }] [\text{disamb by the lawyer}]$

- First pass: Disamb – reduced slower than unreduced
I. The Animate/Inanimate \( V \) examined \( [\text{disamb by the lawyer}] \)

II. The Animate/Inanimate that was \( V \) examined \( [\text{disamb by the lawyer}] \)

- First pass: Disamb – reduced slower than unreduced

\( \Rightarrow \) incremental interpretation does not affect parsing
Altmann and Steedman, (1988): NP/VP-attachment ambiguity

Context affects syntactic attachment

Target sentence:

(10) The burglar blew open the safe with...
Altmann and Steedman, (1988): NP/VP-attachment ambiguity

Context affects syntactic attachment

Target sentence:

(10) The burglar blew open the safe with...
    a. ...the dynamite.
    b. ...the new lock.
Altmann and Steedman, (1988): NP/VP-attachment ambiguity

Context affects syntactic attachment

Target sentence:

(10) The burglar blew open the safe with...
    a. ...the dynamite.
    b. ...the new lock.

(11) a. The burglar blew open $[^{NP} \text{the safe with the N}]$
    b. The burglar $[^{VP} \text{blew open} \ [^{NP} \text{the safe}] \text{ with the N }]$

- more than one safe $\rightarrow$ NP-attachment supported
- just one safe $\rightarrow$ VP-attachment supported
Altmann et al. (1988): context influence

(12) NP-attachment support
a. A burglar broke into a bank carrying some dynamite. He planned to blow up a safe. Once inside he saw that there was a safe which had a new lock and a safe which had an old lock.

(13) VP-attachment support
a. A burglar broke into a bank carrying some dynamite. He planned to blow up a safe. Once inside he saw that there was a safe which had a new lock and a strongbox which had an old lock.
Altmann et al. (1988): context influence

(12) NP-attachment support
a. A burglar broke into a bank carrying some dynamite. He planned to blow up a safe. Once inside he saw that there was a safe which had a new lock and a safe which had an old lock.

(13) VP-attachment support
a. A burglar broke into a bank carrying some dynamite. He planned to blow up a safe. Once inside he saw that there was a safe which had a new lock and a strongbox which had an old lock.

The burglar blew open the safe with the dynamite (new lock) and...
Altmann et al. (1988): results
Altmann et al. (1988): results
NP-attachment >> VP-attachment

Late Closure:
if more than one phrase is compatible with the phrase P, parser attaches P to the most local phrase currently being processed

Frazier (1978)

(14) Jessie put the book Kathy was reading in the library...
Altmann et al. (1988)

explanation of results

(NP-attachment >> VP-attachment) \times \text{context}

Principle of Parsimony:
A reading that carries fewer unsatisfied presuppositions will be favored over one that carries more.

Steedman, 2001

(15) The burglar blew open the safe with the ...
(16) Put the apple on the towel.
    a. Put $[NP \text{ the apple on the towel }]$
    b. $[VP \text{ Put } [NP \text{ the apple }] \text{ on the towel } ]$

Experimental sentences:

(17) Put the apple on the towel in the box. (Locally Ambiguous)

(18) Put the apple that’s on the towel in the box. (Locally Unambiguous)
Tanenhaus et al. (1995): visual context
Tanenhaus et al. (1995): visual context

(19)  

a. Put the apple on the towel in the box.  
    (Ambiguous)  

b. Put the apple that’s on the towel in the box.  
    (Unambiguous)
Tanenhaus et al. (1995)

explanation of results

(NP-attachment >> VP-attachment) × context

Principle of Parsimony:
A reading that carries fewer unsatisfied presuppositions will be favored over one that carries more. Steedman, 2001

(20) Put the apple on the towel in the box.
Interim summary

- Semantic information available to parser
- Semantic constraints incrementally used
  Altmann and Steedman, 1988; Tanenhaus et al., 1995
- Semantic constraints not incrementally used
  Ferreira and Clifton, 1986
Interaction of semantic selectional restrictions and parsing

(21) a. The defendant examined by the lawyer turned out to be unreliable. (Animate Reduced)
b. The defendant that was examined by the lawyer turned out to be unreliable. (Animate Unreduced)
Trueswell et al. (1994)

Interaction of semantic selectional restrictions and parsing

(21)  a. The defendant examined by the lawyer turned out to be unreliable.  (Animate Reduced)
b. The defendant that was examined by the lawyer turned out to be unreliable.  (Animate Unreduced)

(22)  a. The evidence examined by the lawyer turned out to be unreliable.  (Inanimate Reduced)
b. The evidence that was examined by the lawyer turned out to be unreliable.  (Inanimate Unreduced)
Trueswell et al. (1994): results

![Graph showing reading times in milliseconds for different scoring regions and conditions.](image-url)
Trueswell et al. (1994): results

![Graph showing reading times in msec for different scoring regions and conditions.](image)
Eye-tracking & Trueswell et al. (1994)

explanation of results

• Main-clause >> reduced-relative clause (if both possible)
• Parser uses lexical information (selectional restrictions) to change preferences
• But what about Ferreira and Clifton, 1986?
• Ferreira and Clifton, 1986 – inanimate condition not always violating selectional restrictions:
Eye-tracking & Trueswell et al. (1994)

explanation of results

• Main-clause >> reduced-relative clause (if both possible)
• Parser uses lexical information (selectional restrictions) to change preferences
• But what about Ferreira and Clifton, 1986?
• Ferreira and Clifton, 1986 – inanimate condition not always violating selectional restrictions:
The car towed from the parking lot...
The meal brought to the highest priest...
Summary

• Evidence that lexical and contextual semantics can guide parser in syntax
  Altmann and Steedman, 1988; Marslen-Wilson, 1973, 1975; Tanenhaus et al., 1995; Trueswell et al., 1994

• Consequences?
Setting the stage

Theoretical considerations & parsing

Appendix: developments & refinements
Theoretical considerations

- Semantics (compositionality principle)
- Parsing (strong competence and rule-to-rule assumption)
- Parsers
Semantics – interpretation is productive

- We can interpret novel, previously unheard of sentences

Consequences:
- Interpretation is rule based (as opposed to memorized)
- Interpretation proceeds by building bigger blocks out of smaller blocks

What are the smaller blocks? What are the bigger blocks?
→ Constituents
Understanding productivity

Principle of compositionality

The meaning of a complex expression is fully determined by its structure (syntax) and the meaning of its constituents (parts)

Stanford Encyclopedia of Philosophy
Assumptions about parsing

**Strong competence (Bresnan and Kaplan, 1982)**
There exists a direct correspondence between the rules of a grammar and the operations performed by the human language processor.

**Rule-to-rule compositionality (Montague, 1973)**
Each syntactic rule corresponds to a rule of semantic interpretation. (⇒ entities combined by syntactic rules must be semantically interpretable)
Grammar

I. \( S \rightarrow NP \ VP \)

II. \( NP \rightarrow D \ N \)

III. \( NP \rightarrow NP \ VP \)

   the: D

   book, opinion: N

IV. \( PP \rightarrow P \ NP \)

V. \( VP \rightarrow V \ PP \)

VI. \( VP \rightarrow V \)

fell: V
Grammar

I. \( S \rightarrow \text{NP VP} \)

II. \( \text{NP} \rightarrow \text{D N} \)

III. \( \text{NP} \rightarrow \text{NP VP} \)

the: D

book, opinion: N

IV. \( \text{PP} \rightarrow \text{P NP} \)

V. \( \text{VP} \rightarrow \text{V PP} \)

VI. \( \text{VP} \rightarrow \text{V} \)

fell: V

\[
\begin{array}{c}
\text{S} \\
\text{NP} \quad \text{VP} \\
\text{D} \quad \text{N} \quad \text{V} \\
\text{the} \quad \text{book} \quad \text{fell}
\end{array}
\]
Parser: top-down

I. S → NP VP

II. NP → D N

III. NP → NP VP

IV. PP → P NP

V. VP → V PP

VI. VP → V

- **expand**: if the stack shows a symbol \( X \) on top, and the grammar contains a rule \( X \rightarrow \alpha \) then replace the stack symbol \( X \) with the sequence of symbols \( \alpha \)

- **scan**: if the stack shows one of the grammar’s terminal symbols \( Y \) on top, and \( w \), the current word being parsed, is of category \( Y \), then remove \( w \) from the input and \( Y \) from the stack

J. T. Hale, 2014
import pyactr as actr

environment = actr.Environment(focus_position=(320, 180))

actr.chunktype("parsing_goal", "stack_top stack_middle\nstack_bottom parsed_word task")
actr.chunktype("parse_state", "mother daughter1 daughter2")
actr.chunktype("word", "form cat")

parser = actr.ACTRModel(environment)
dm = parser.decmem
g = parser.goal
imaginal = parser.set_goal(name="imaginal", delay=0.05)
ACT-R TD parser; add lexical information into decl. memory

```python
dm.add(actr.chunkstring(string=""
    isa word
    form evidence
    cat N
"
))

g.add(actr.chunkstring(string=""
    isa parsing_goal
    task read_word
    stack_top S
"
))
```
ACT-R TD parser; add rules

```plaintext
parser.productionstring(name="encode word", string=""
    =g>
    isa            parsing_goal
    task           read_word
    =visual>
    isa            _visual
    value          =val
    =>
    =g>
    isa            parsing_goal
    task           get_word_cat
    parsed_word    =val
    ~visual>
    """"}
```
parser.productionstring(name="retrieve category", string=""
   =g>
  isa                        parsing_goal
  task                      get_word_cat
  parsed_word               =w
  ==>                       +retrieval>
  isa                        word
  form                      =w
  =g>
  isa                        parsing_goal
  task                      match_category

  """)
ACT-R TD parser; add rules

```java
parser.productionstring(name="match category", string="""
   =g>
  isa      parsing_goal
  task     match_category
  ?retrieval>
  state    free
  buffer   full
  =retrieval>
  isa      word
  cat      =c
  ==>      
  =g>
  isa      parsing_goal
  task     parsing
  parsed_word =c
"")
```
ACT-R TD parser; add rules

```
parser.productionstring(name="expand: S ==> NP VP", string=""
  =g>
  isa  parsing_goal
  task  parsing
  stack_top  S
  stack_middle  =s2
  ==>  
  =g>
  isa  parsing_goal
  stack_top  NP
  stack_middle  VP
  stack_bottom  =s2
  +imaginal>
  isa  parse_state
  mother  S
  daughter1  NP
  daughter2  VP
  """"
```
parser.productionstring(name="expand: NP ==> D N", string=""
  =g>
  isa        parsing_goal
  task       parsing
  stack_top  NP
  stack_middle  =s2
  ==> 
  =g>
  isa        parsing_goal
  stack_top  D
  stack_middle  N
  stack_bottom  =s2
  +imaginal>
  isa        parse_state
  mother    NP
  daughter1  D
  daughter2  N
  """"
ACT-R TD parser; add rules

```
parser.productionstring(name="scan: word", string=""
    =g>
    isa     parsing_goal
    task    parsing
    stack_top =y
    stack_middle =x
    stack_bottom =b
    parsed_word =y
    ==> =g>
    =g>
    isa     parsing_goal
    task    press_space
    stack_top =x
    stack_middle =b
    stack_bottom None
    parsed_word None
"")
```
ACT-R TD parser; add rules

```python
parser.productionstring(name="press spacebar", string=""
  =g>
  isa      parsing_goal
  task     press_space
  stack_top ~None
  ?manual>
  state    free
  ==>      
  =g>
  isa      parsing_goal
  task     read_word
  +manual>
  isa      _manual
  cmd      'press_key'
  key      'space'
  ~imaginal>
  ""
)```
Parser: bottom-up

I. $S \rightarrow NP \ VP$

II. $NP \rightarrow D \ N$

III. $NP \rightarrow NP \ VP$

IV. $PP \rightarrow P \ NP$

V. $VP \rightarrow V \ PP$

VI. $VP \rightarrow V$

- **reduce**: if the top of the stack shows a sequence of symbols $\alpha$, and there is a grammar rule $X \rightarrow \alpha$, then replace $\alpha$ on the stack with $X$.
- **shift**: if the current word of the sentence is $w$, push $w$ on to the top of the stack.

J. T. Hale, 2014
environment = actr.Environment(focus_position=(320, 180))

actr.chunktype("parsing_goal", "stack_1 stack_2 stack_3 stack_4 stack_5 parsed_word task")
actr.chunktype("parse_state", "mother daughter1 daughter2")
actr.chunktype("word", "form cat")

parser = actr.ACTRModel(environment)
dm = parser.decmem
g = parser.goal
imaginal = parser.set_goal(name="imaginal", delay=0.05)
 ACT-R BU parser; add rules

`parser.productionstring(name="shift word and project it", string=""

=g>
isa parsing_goal

=retreival>
isa word
cat =y

=retreival>
isa parsing_goal

=g>

ACT-R BU parser; add rules

```
parser.productionstring(name="reduce: NP ==> D N", string=""

  =g>
  isa        parsing_goal
  task       parsing
  stack_1    N
  stack_2    D
  stack_3    =s3
  stack_4    =s4
  stack_5    =s5
  ==>        =g>
  =g>
  isa        parsing_goal
  stack_1    NP
  stack_2    =s3
  stack_3    =s4
  stack_4    =s5
  stack_5    None
+imaginal>
  isa        parse_state
  mother     NP
  daughter1  D
  daughter2  N"")
```
parser.productionstring(name="press spacebar", string="""
    =g>
    isa parsing_goal
    task parsing
?manual>
    state free
?imaginal>
    state free
==> =g>
    isa parsing_goal
    task read_word
+manual>
    isa _manual
    cmd 'press_key'
    key 'space'
~imaginal>
""", utility=-10)
Parser: left-corner (eager)

I.  $S \rightarrow \text{NP VP}$

II. $\text{NP} \rightarrow \text{D N}$

III. $\text{NP} \rightarrow \text{NP VP}$

- **project**: if the top of the stack is a symbol $Y$, and there is a grammar rule $X \rightarrow Y \beta$ whose right-hand side starts with $Y$, then replace $Y$ with new symbols: an expectation for each of the remaining righthand side symbols, and a record that $X$ has been found

- **project+complete**: if the top of the stack is $Y$, and right below it is an expectation $[X]$, then replace both with the remaining expectations $\beta$

- **shift**: if the current word of the sentence is $w$, push $w$ on to the top of the stack.

Resnik, 1992
ACT-R LC parser; add rules

```python
def parser.productionstring(name="project and complete: NP ==> D N", string=True)
    isa stack_1 D
    isa stack_2 NP
    isa stack_3 =s3
    isa stack_4 =s4
    isa parsing_goal
    isa parse_state
    return
```

```
+imaginal>
    isa mother NP
    isa daughter1 D
    isa daughter2 N
    ""
```
parser.productionstring(name="project: NP ==> D N", string=""

=g>
isa parsing_goal
stack_1 D
stack_2 =s2
stack_2 ~NP
stack_3 =s3
stack_4 =s4
==>
=g>
isa parsing_goal
stack_1 N
stack_2 NP
stack_3 =s2
stack_4 =s3
+imaginal>
isa parse_state
mother NP
daughter1 D
daughter2 N
"")
Top-down:

- **expand**: if the stack shows a symbol $X$ on top, and the grammar contains a rule $X \to \alpha$ then replace the stack symbol $X$ with the sequence of symbols $\alpha$

- **scan**: if the stack shows one of the grammar’s terminal symbols $Y$ on top, and $w$, the current word being parsed, is of category $Y$, then remove $w$ from the input and $Y$ from the stack

Bottom-up:

- **reduce**: if the top of the stack shows a sequence of symbols $\alpha$, and there is a grammar rule $X \to \alpha$, then replace $\alpha$ on the stack with $X$.

- **shift**: if the current word of the sentence is $w$, push $w$ on to the top of the stack.

Left-corner (eager):

- **project**: if the top of the stack is a symbol $Y$, and there is a grammar rule $X \to Y \beta$ whose right-hand side starts with $Y$, then replace $Y$ with new symbols: an expectation for each of the remaining right-hand side symbols, and a record that $X$ has been found

- **project+complete**: if the top of the stack is $Y$, and right below it is an expectation $[X]$, then replace both with the remaining expectations $\beta$

- **shift**: if the current word of the sentence is $w$, push $w$ on to the top of the stack.
Parsers and interpretation

I. $S \rightarrow NP \ VP$

II. $NP \rightarrow D \ N$

III. $NP \rightarrow NP \ VP$

IV. $PP \rightarrow P \ NP$

V. $VP \rightarrow V \ PP$

VI. $VP \rightarrow V \ NP$

(23) The evidence examined by the doctor...
Summary

I. Bottom-up parsing coupled with compositionality predicts (the effect of) incremental interpretation too late

II. Top-down & left-corner parsing (with the standard theory of adjunction) predict that the decision happens before the disambiguating incremental interpretation

III. For top-down & left-corner parsers: Incremental interpretation has to be able to interpret incomplete constituents
Setting the stage

Theoretical considerations & parsing

Appendix: developments & refinements
Solution I: Steedman, 2001

- bottom-up parsing using Combinatory Categorial Grammar
- only constituents (well-formed syntactic objects) receive interpretation
- Interpretation becomes incremental due to extra composition rules
Notation of bottom-up parser

the: D  lawyer: N  examined: VBD  the: D  evidence: N
NP → D N  NP → D N
___   ___
  NP  NP

VP → VBD NP  VP → VBD NP
___   ___
  VP  VP

S → NP VP  S → NP VP
___   ___
  S  S
Bottom-up parsing rule

Current rule
- If you have evidence for A and B and you have a rule \( X \rightarrow A \ B \), postulate X

Extra rule (rule composition)
- Two rules: \( X \rightarrow A \ B \), \( B \rightarrow C \ Y \)
- You have evidence for A and C (being empty counts as evidence)
- Postulate a new rule, \( X \rightarrow Y \)
Bottom-up parser with extra rule

the:D lawyer:N examined:VBD the:D evidence:N
NP → D N

NP
S → NP VP VP → VBD NP

S → NP VP → VBD NP

S → NP NP → D N

S → N

S
Bottom-up parser with extra rule

The parser can parse incrementally:

(24) The woman that John saw...

The parser cannot parse incrementally:

(25) The woman that every man saw...

Demberg, 2012

Coordination as a constituency test:

(26) [ books that every ] and [ journals that no ] accordionist liked (?)
Solution II: Interpreting non-constituents is valid
Stabler, Shieber and Johnson

(27) The evidence examined...

- (27) is a non-constituent, but it can be interpreted (why should it not?)
- Dropping the strict mapping between syntax and semantics – enough for top-down and left-corner parser
- On bottom-up parser and adjunction: Shieber and Johnson, 1993
Asynchronous processing

(a) – asynchronous
(b) – synchronous
(b) – in some conditions disadvantaged:
\[ x = 1, \text{ or } y = 0 \]

Circuit for computing: \[ z = xy + (-y) \]
Summary

- top-down parsing, bottom-up parsing, left-corner parsing
- incremental interpretation and the limits with bottom-up parsing
- incremental interpretation and the interpretation of non-constituents


References II


