Cognitive Modeling

Lecture 8: Models of Syntactic Processing

Sharon Goldwater

School of Informatics
University of Edinburgh
sgwater@inf.ed.ac.uk

February 4, 2010

Reading: Cooper (2002: Ch. 7).

Incrementality

Parsing: extracting syntactic structure from a string; prerequisite for assigning a meaning to the string.

The sentence processor builds structures *incrementally* (word by word) as the input comes in (Tanenhaus et al. 1995).

This can lead to *local ambiguity.*

Example:

(1) The athlete realized his potential . . .
  a. . . . at the competition.
  b. . . . would make him a world-class sprinter.

Garden Paths

- *Early commitment:* when it reaches *potential*, the processor has to decide which structure to build.
- If the parser makes the wrong choice (e.g., NP reading for sentence (1-b)) it needs to backtrack and revise the structure.
- A *garden path* occurs, which typically results in longer reading times (and reverse eye-movements).
- Some garden paths are so strong that they parser fails to recover from them.

Garden Paths

More examples of garden paths:

(2)
a. I convinced her children are noisy.
b. Until the police arrest the drug dealers controlled the street.
c. The old man the boat.
d. We painted the wall with cracks.
e. Fat people eat accumulates.
f. The cotton clothing is usually made of grows in Mississippi.
g. The prime number few.

Dimensions of Parsing

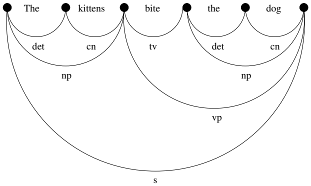
In addition to incrementality, a number of properties are important when designing a model of the HPSM:

- **Directionality:** the parser can process sentence bottom-up (from the words up) or top-down (from the phrase markers down). *Evidence that the HPSM combines both strategies.*
- **Parallelism:** a serial parser maintains only one structure at a time; a parallel parser pursues all possible structures. *Controversial issue; evidence for both serialism and limited parallelism.*
- **Interactivity:** the parser can be encapsulated (only access to syntactic information) or interactive (access to semantic information, context). *Evidence for limited interactivity.*

A Bottom-Up Parallel Parser

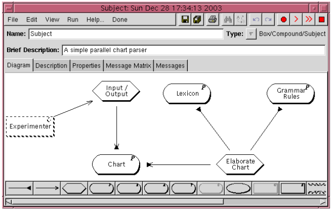
The parser constructs a *chart,* a compact representation of all the analyses of a sentence.

Goal: find an S edge that spans the whole sentence. Example:



A Bottom-Up Parallel Parser

Architecture of a simple parser that constructs the chart bottom-up:



Chart, Lexicon, Grammar Rules

- Chart edges are represented as predicates of the form:

  `edge(LeftVertex,RightVertex,Content,Level)`

  - `LeftVertex` and `RightVertex` are vertex indices
  - `Content` is the content of the edge (e.g., `word(cat)`)
  - `Level` is formatting information (not discussed here)
- Examples for items in the lexicon:

  `category(the,det)`
  `category(kittens,cn)`
- Examples for grammar rules:

  `rule(s,[np,vp])`
  `rule(np,[pn])`

Input/Output Process

Rule 1: Add a word to the first position of the chart:

TRIGGER `word(W)`
IF not `edge(_,_,_,_)` is in **Chart**
THEN add `edge(0,1,word(W),0)` to **Chart**

Rule 2: Add a word to the next position of the chart:

TRIGGER `word(W)`
IF `edge(N0,N1,word(W1),Y)` is in **Chart**
not `edge(N1,N2,word(W2),Y)` is in **Chart**
`N2` is `N1 + 1`
THEN add `edge(N1,N2,word(W),Y)` to **Chart**

Elaborate Chart Process

Rule 1: Lexical look-up:

IF `edge(N0,N1,word(W),L1)` is in **Chart**
`category(W,C)` is in **Lexicon**
`L` is `L1 + 1`
THEN add `edge(N0,N1,cat(C),L)` to **Chart**

Rule 2: Apply unary grammar rules:

IF `edge(N0,N1,cat(C1),L1)` is in **Chart**
`rule(C,[C1])` is in **Grammar Rules**
`L` is `L1 + 1`
THEN add `edge(N0,N1,cat(C),L)` to **Chart**

Elaborate Chart Process

Rule 3: Apply binary grammar rules:

IF `edge(N0,N1,cat(C1),L1)` is in **Chart**
`edge(N1,N2,cat(C2),L2)` is in **Chart**
`rule(C,[C1,C2])` is in **Grammar Rules**
`L` is `max(L1,L2) + 1`
THEN add `edge(N0,N2,cat(C),L)` to **Chart**

Similar rules for grammar rules with more than two categories.

Properties of the Model

Simple, but complete chart parser with the following properties:

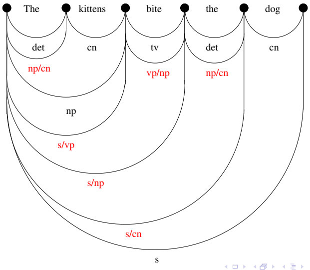
- **bottom-up:** parsing is driven by the addition of words to the chart; chart is expended upwards from lexical to phrasal categories;
- **limited incrementality:** when a new word appears, all possible edges are added to the chart; then the system quiesces and waits for the next word;
- **parallelism:** all chart edges are added at the same time (default Cogent behavior); multiple analyses are pursued.

Left Corner Parsing

Bottom-up parsing processes each word as it appears, but may have unconnected structure. *Left corner parsing* is more cognitively plausible: each word is integrated into the structure as it appears.

- The chart of a left-corner parser contains *active edges*: incomplete constituents that represent predictions about what will come next.
  - Ex: NP/CN is a constituent that lacks a CN in order to become an NP.
- For a completed edge Y and a grammar rule $X \rightarrow Y\ Z$, introduce the active edge X/Z into the chart, where Y and X/Z span the same part of the string.

Example of a Left Corner Chart



Serial Parsing

If parsing was fully parallel, all analyses of a sentence would be equally available; there would be no garden paths.

In the literature, two types of models have been assumed:

- *ranked parallel:* multiple structures are pursued in parallel; they are ranked in order of preferences; garden paths occur if a low-ranked structure turns out to be correct;
- *serial:* only one structure is pursued; if it turns out to be incorrect, then a garden path occurs.

Serial Parsing

Serial left-corner parser with backtracking:

- At each point of ambiguity, the parser has to chose one structure;
- if the structure turns out to be incorrect; the parser has to backtrack;
- at the last point of ambiguity, the incorrect structure is disassembled, and another alternative is pursued instead.

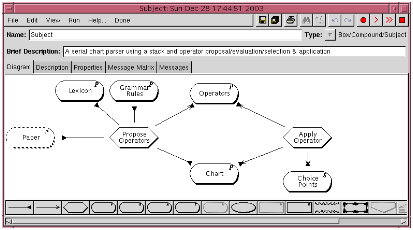
A Serial Model of Left Corner Parsing

Computational requirements:

- **operator selection:** each stage of processing, the parser has to select what to do: elaborate the current structure, read the next word, backtrack;
- **dept-first search:** pursue a structure as far as possible before alternatives are considered; requires inhibition of some edges in the chart;
- **backtracking:** previous states of the parser must be recoverable if backtracking occurs; requires removing edges from the chart.

A Serial Model of Left Corner Parsing

Basic architecture:



Propose Operators Process

This process can propose the following operators:

- reading the next word: `add_word`;
- lexical lookup: `add_edge`;
- project up from completed categories to parent categories (i.e., create active edge): `add_edge`;
- merge active edge with following edge (i.e., create passive edge): `add_edge`.

For details see Cooper (2002: p. 307).

Apply Operator Process

Rule 1: Select operator with the highest evaluation:

IF operator(Operator,value(Score)) is in **Operators**
  not operator(AnyOp,selected) is in **Operators**
  not operator(OtherOp,value(OtherScore)) is in **Operators**
    OtherScore is greater than Score
THEN delete operator(Operator,value(Score)) from **Operators**
  add operator(Operator,selected) to **Operators**

Rule 2: Apply the selected operator, remove all others:

IF operator(Operator,selected) is in **Operators**
THEN delete all operator(_,_) from **Operators**
  add operator(Operator,apply) to **Operators**

Apply Operator Process

Rule 3: Push unselected operators onto the stack (note: this rules fires in parallel with Rule 2):

IF operator(Operator,selected) is in **Operators**
  Ops is the list of all operator(O,value(V)) such that
    operator(O,value(V)) is in **Operators**
    V is greater than 0
  Ops is distinct from []
  get_context(Context)
THEN send push(Choices(Context,Ops)) to **Choice Points**

Rule 4: Remove applied operators:

IF operator(Operator,apply) is in **Operators**
THEN delete operator(Operator,apply) from **Operators**

Apply Operator Process

Rule 5: Add a word to the next position in the chart:

IF operator(add_word(W,apply)) is in **Operators**
  get_word_position_parameters(N0,N1)
  get_context(TS)
THEN add edge(N0,N1,word(W),W,0,TS) to **Chart**

Rule 6: Add an edge of the specified type to the chart

IF operator(add_edge(N0,N1,C,S,L),apply) is in **Operators**
  get_context(TS)
THEN add edge(N0,N1,cat(C),S,L,TS) to **Chart**

Also required: rules for backtracking (Cooper 2002: p. 307).

Properties of the Model

Properties of the left corner model:

- this model will parse garden path sentences such as *the horse raced past the barn fell;*
- extensive backtracking will occur for such sentences; only possible if the stack size of the choice point stack is sufficient;

Potential problems:

- backtracking requires that parse failure is detected; requires that the parser knows where the sentence boundaries are;
- operator evaluations are fixed; context or experience is not taken into account; no attempt to minimize backtracking.

Summary

- The human parser builds syntactic structure in response to strings of words;
- parsing models have to capture the incrementality of human parsing and account for ambiguity resolution (garden paths);
- parsing models can be implemented in Cogent using a chart (representing partial syntactic structure);
- left-corner parsing models achieves full incrementality;
- employs operator selection to model serial parsing and backtracking.

References

Cooper, Richard P. 2002. *Modelling High-Level Cognitive Processes*. Lawrence Erlbaum Associates, Mahwah, NJ.

Tanenhaus, Michael K., Michael J. Spivey-Knowlton, Kathleen M. Eberhard, and Julie C. Sedivy. 1995. Integration of visual and linguistic information in spoken language comprehension. *Science* 268:1632–1634.