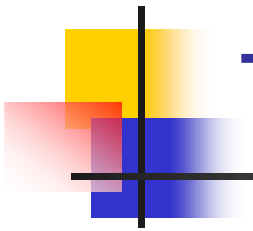
LG-Soar: Parsing for information

Deryle Lonsdale, Merrill Hutchison,
Tim Richards, William Taysom

(The BYU NL-Soar Research Group)



The challenge

- Mine content from problematic text
- Address complicated linguistic issues
- Output information into a usable format
- Integrate components within an agent architecture

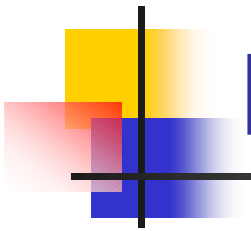
Sample from Savage's text

ABERNETHY, WILLIAM, Wallingford, m. 1673 or 4, Sarah, d. of William Doolittle, had William, and Samuel, and d. 1718, when his two s. admin. on his est. Early this name was writ. Ebenetha, or Abbenatha, acc. Hinman; but in mod. days the descend. use the spell. here giv.

ABINGTON, WILLIAM, Maine, 1642. Coffin.

ABORNE. See Eborne.

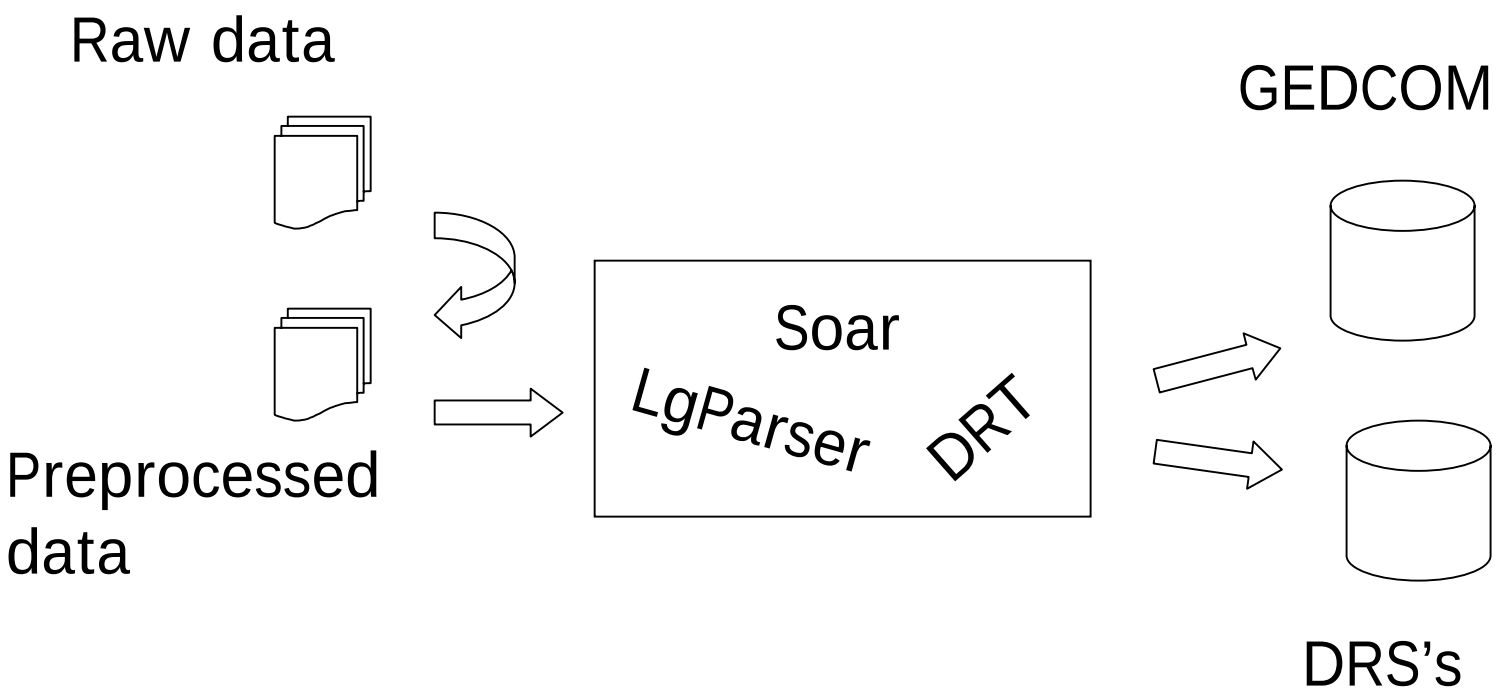
ACRERLY, ACCORLEY, or ACRELY, HENRY, New Haven 1640, Stamford 1641 to 53, Greenwich 1656, d. at S. 17 June 1668, wh. is the date of his will. His wid. Ann, was 75 yrs. old in 1662. Haz. II. 246. ROBERT, Brookhaven, L. I. 1655, adm. freem. of Conn. jurisdict. 1664. See Trumbull, Col. Rec. I. 341,428. SAMUEL, Brookhaven, 1655, perhaps br. of the preced.

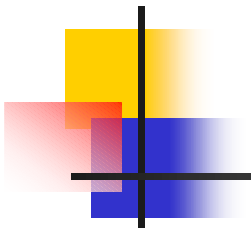


LG-Soar

- Integration of three major components
  - Regular-expression-based text preprocessing
  - The Link Grammar parser (Sleator & Temperley)
  - The Soar architecture
- Robust, versatile text processing engine useful for difficult-to-handle input
- Why a new Soar-based parser? NL-Soar is:
  - Designed for cognitive modeling of natural language use
  - Not (yet) versatile enough to handle grammatically problematic text

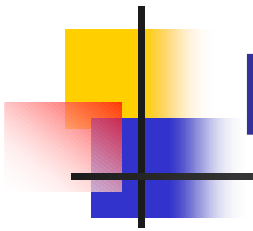
The system





Initial stage: preprocessing

- Goals:
  - Regularize, standardize nonstandard input text
  - Explicitize predictable implicit relationships
  - Locate, reformat individual entries
  - Replace abbreviations (ambiguous and not)
- Reason: facilitate subsequent linguistic processing
- Method: comprehensive set of subroutines and regular expressions coded in Perl
- Result: cleaned-up input text



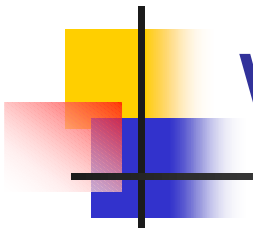
Example preprocessing

- Input

```
HAVERHILL, THOMAS, m. at Andover 6 Jan. 1659, Unice Singletary
of Salisbury. Was freem. 1666. Was k. by the Ind. 15 Mar. 1698.
```

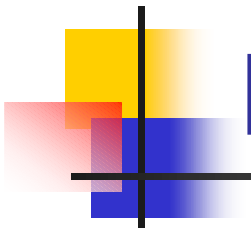
- Output

```
Thomas Haverhill,  married at Andover 6 January 1659,
Unice Singletary of Salisbury.**

Was freeman 1666.**

Was killed by the Indians 15 March 1698.**
```



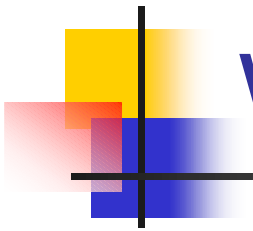
Why Soar?

- Flexible multipurpose platform
  - Goal-directed problem solving
  - Agent-based architecture: web search
  - Already used successfully in other applications
- Used in other difficult parsing tasks
  - NL-Soar for modeling language use in humans
  - Representing and tracking referents in disourse
- Potential for further application
  - Machine learning



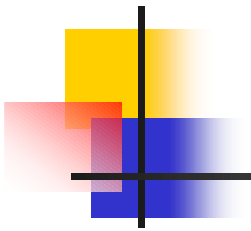
LG-Soar input/output

- Input: fairly clean (if not completely grammatical) textual input
  - i.e. Preprocessed text as described previously
- Output: some representation of structure that will allow for the next stage of processing
  - Traditional linguistic structures are often cumbersome



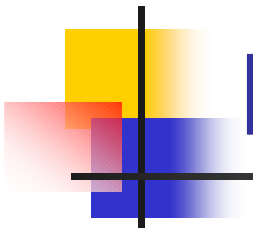
Why the LG parser?

- Freely available for research purposes
- Robust
- Simple, explicit relations for next stage of processing
- Fast
- Written in C
- More appropriate for the task than traditional phrase-structure grammars



Integration of Soar/LG engines

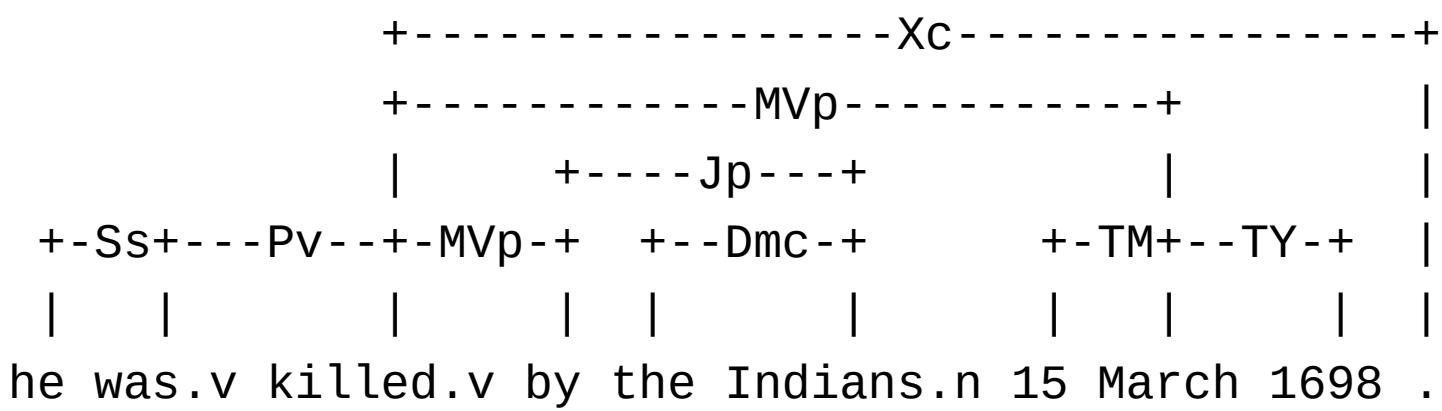
- Soar and LG engine both use C at the lowest levels
- Tcl is used for higher-level funtions
- Tcl acts as “glue” between Link Grammar engine and Soar engine
- RESULT: LG-Soar, including Tcl commands that call Link Grammar functions and pass information into the basic Soar processor



Exploring Link Grammar

- What is a link?
  - Two parts, + and –
  - Shows a relationship between pairs of words
    - Subject + verb
    - Verb + object
    - Preposition + object
    - Adjective + adverbial modifier
    - Auxiliary + main verb
  - Labels each relationship
- Potential links are specified by technical rules
- Possible to score linkages, penalize links

Sample link parse



Sample LG rule entries

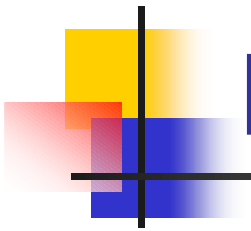
words/words.y: % year numbers

NN+ or NIa- or AN+ or MV- or ((Xd- & TY- & Xc+) or TY-)
or ({EN- or NIc-} & (ND+ or OD- or ({{@L+} & DD-} &
([[Dmcn+]] or ((<noun-sub-xnoappositive> or TA-) & (JT- or IN-
or <noun-main-xnoyear>))))));

<vc-fill>: ((K+ & {[[@MV+]]} & O*n+) or ({O+ or B-} & {K+}) or
[[@MV+ & {Xc+} & O*n+]]) & {Xc+} & {@MV+};

LG parser's robustness

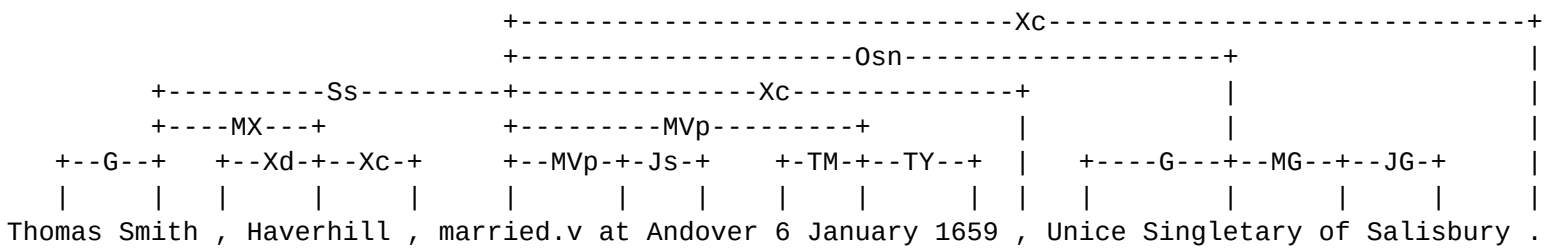
```
Mary married I think, 23 November 1661, Samuel Gay.
No complete linkages found.

          +-------------------------Xc-------------------------+
          +-----------------------Osn-----------------------+  |
          +------------------Xc------------------+          |  |
          +-------------MVp------------+         |          |  |
  +--Ss--+                         +--TM-+--TY--+  |    +--G-+  |
  |      |                         |     |      |  |    |    |  |
Mary married.v [I] [think] [,] 23 November 1661 , Samuel Gay .
```

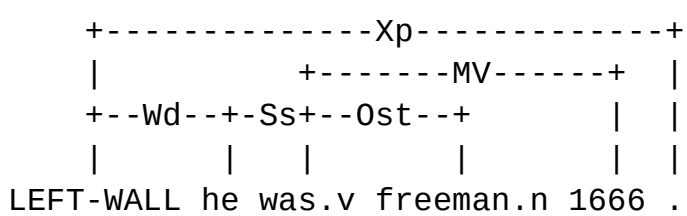


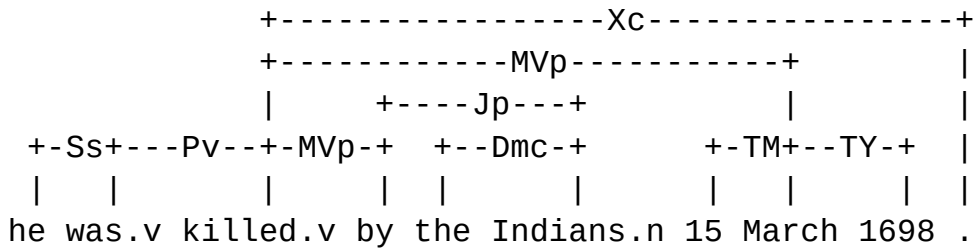
Enhancing the grammar

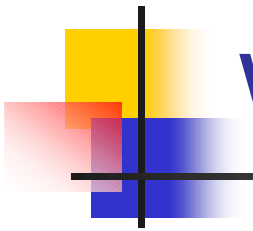
- Order w/in dates (e.g. May 24 or 24 May)
- Expand range recognizable years
- Allow year postmodifiers (e.g. died May 1655)
- Allow comma-separated verbal arguments (e.g. married 6 May 1694, Ann Lynde)
- Penalize months, years as appositives (e.g. died of smallpox, 24 October 1678)
- Allow telegraphy (e.g. He was son of Joe.)
- Add lexicon entries (e.g. freeman)

LG example parses



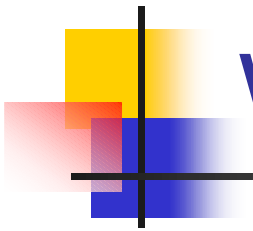






What DRT?

- Discourse Representation Theory
- A particular way of dealing with semantics and logic in natural language
- Goals:
  - To represent utterances in a way that emphasizes their logical structure
  - To allow language processing of phenomena that depend on logical structure.
- Data types: Discourse representation structures (DRS's)



What makes up a DRS?

- Discourse referents:
  - Variables, representing objects; anything which can serve as the antecedent for an anaphor
- Conditions:
  - Represent properties and relationships
- Examples:

```
Thomas(u)
v married w
```

- Processing: links → protoDRS → DRS

Associating information

- Individuals: i1, i2

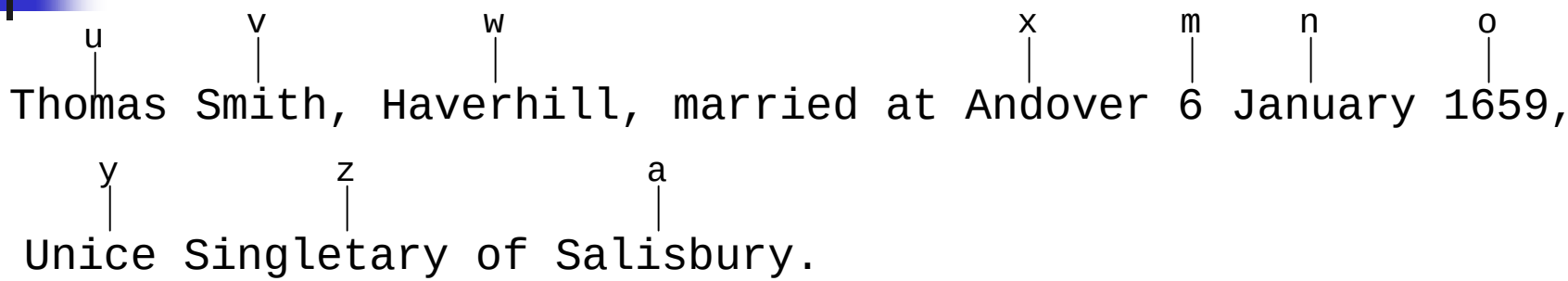
i1

```
Name: Thomas Smith
Lived: Haverhill
Married: i2
-where? Andover
-when? 6 January 1659
Died: 15 March 1698
```

i2

```
Name: Unice Singletary
Lived: Salisbury
```

Proto-DRS



Thomas(u), Smith(v), Haverhill(w), Andover(x), 6(m), January(n),

modifier="Haverhill"
propername=uv

time(day m, month n), 1659(o), time(month n, year o), Salisbury(a),

Unice(y), Singletary(z), prep("at", x), verbal("married", v, x)

propername=yz

modifier="Andover"
modifier="January"

Corresponding DRS

u, v, w, x, y, z, a, b, c, d, e

Thomas(u), Smith(v), Haverhill(w), v~w,

propername=uv

Andover(x), Unice(y), Singletary(z),

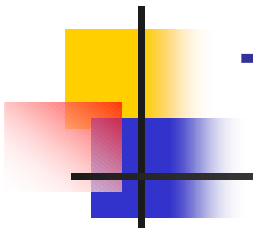
of=a
propername=yz

Salisbury(a), v married z, b=v, freeman(c),

at x
day=6
month=January
year=1659

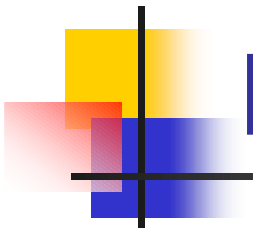
b was c, d=v, Indians(e), d was killed

by e
day=15
month=March
year=1698



The output

- Predicate-argument relationships
- Discourse representation structures
  - CLIG grapher output
- GEDCOM files



Future work and applications

- More DRT implementation
- Other types of unstructured data
- Other languages
- Semistructured text
- Integration with lexical resources (e.g. WordNet, onomastica)
- Machine learning