Declarative Syntactic Processing of Natural Language Using Concurrent Constraint Programming and Probabilistic Dependency Modeling

Irene Langkilde-Geary
Natural Language Technology Group
University of Brighton
The Problem

- Improve the quality of MT output using syntactic modeling

- Sampling of efforts
  - Charniak et al., 2003
  - Och et al., 2004
  - Galley et al., 2006
  - Has-san et al., 2007
  - Chang and Toutanova, 2006
  - Koehn and Hoang, 2007
Challenges

- Annotation for training
- Compatibility of representation across components
- Interdependence of realization subtasks, i.e.,
  - lexical choice,
  - choice of syntactic structure,
  - word order,
  - etc.
- Combinatorial complexity and consequent computational (in)tractability
The Need for Declarativeness

- Suggested by an accumulation of work related to generation system architectures
  - Cahill et al., 2000
  - Calder et al., 1999
  - Beale et al., 1998
  - Beale, 1997
  - Elhadad et al., 1997
  - Smedt et al., 1996
  - Robin, 1994
  - Meteer, 1990

- as well as own experience trying to integrate syntactic modeling into HALogen
Generation System Architectures

- Sequential (pipeline)
- Integrated
- Interactive (feedback)
- Revision-based
- Blackboard/Whiteboard
What does “declarative” mean?

- Representations are
  - Stateless
  - Impose no artificial restrictions processing order
- Processing directed by input and inference reasoning mechanisms
Previous work on declarative approaches

- **Linguistics**
  - Theories: eg. HPSG, LFG, TAG, CCG
  - Older work was purely symbolic and had reputation for being brittle and slow
  - More recent work incorporates statistical modeling, but still aims more at parsing than realization, and lacks complete declarativeness

- **Computer Science**
  - Constraint programming
  - . . .
Constraint Programming

- Roots in
  - Artificial Intelligence,
  - Operations Research, and
  - Programming Language design and implementation
- Integrates
  - heterogeneous solvers within
  - general-purpose methodology having strong theoretical foundations
- Facilitates high-level model experimentation
- Designed for finding good average-case solutions to hard combinatorial problems
Our Proposed Approach

- Concurrent CP as an efficient whiteboard architecture for fine-grained declarativeness
  - Mozart/Oz language and programming environment
- Factored dependency-style representation derived from Penn Treebank
  - Langkilde-Geary and Betteridge, LREC 2006
- Combination of
  - hard logical constraint propagation with
  - soft probabilistic optimization
- This paper:
  - Realization and parsing from tokens, not just functional roles
  - Set-based representation of tree structure
Concurrent CP as a Whiteboard

Whiteboard Architecture
- modules
- central repository

Constraint Program
- constraint propagators/solvers
- constraint store
Modeling with CCP

- Variables
- Constraints
  - Basic: individual variable domains
  - Non-basic: Relations between variables
- Search strategy

Methodology
1. Propagate inferences onto var domains
2. Split into complementary subcases
3. Repeat
## CP Formulation: Variables

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>A word id</td>
</tr>
<tr>
<td>HeadID</td>
<td>ID of head word</td>
</tr>
<tr>
<td>Token</td>
<td>Inflected word form</td>
</tr>
<tr>
<td>Role</td>
<td>Syntactic function</td>
</tr>
<tr>
<td>Group type (GT)</td>
<td>Clause, NP, or other</td>
</tr>
<tr>
<td>Direction (DIR)</td>
<td>+/- from head</td>
</tr>
<tr>
<td>Relative Position (RP)</td>
<td>Tree distance from head</td>
</tr>
<tr>
<td>Absolute Position (AP)</td>
<td>Distance from start of sen</td>
</tr>
</tbody>
</table>
CP Formulation (cont): Non-basic Constraints

- **Definitional:**
  - DIR, AP, headID
  - DIR, RP
  - DIR and RP of top node
  - Distinct and sequential sibling RPs (except 0)
  - Distinct and sequential APs
  - AP, RP of siblings

- **Projective tree (AP, RP, ID, headID):**
  - uses set constraints

- **Probabilistic dependency scores**
  - -logprob(feature | history).
# Probabilistic Dependencies

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>INTERDEPENDENCIES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group type</td>
<td><strong>head</strong>: gt</td>
</tr>
<tr>
<td></td>
<td><strong>self</strong>: role, dir, rp, token</td>
</tr>
<tr>
<td>Role</td>
<td><strong>head</strong>: role, gt</td>
</tr>
<tr>
<td></td>
<td><strong>self</strong>: gt, dir, rp, token</td>
</tr>
<tr>
<td>Dir</td>
<td><strong>self</strong>: rp, role, gt</td>
</tr>
<tr>
<td>RP</td>
<td><strong>self</strong>: dir, role, gt</td>
</tr>
</tbody>
</table>
Example: Node Score

*Assume:*
- Given: token, role
- Determination order: dir, rp, gt

\[
\text{NodeScore} = \text{FeatScore}(\text{role}) + \text{FeatScore}(\text{dir}) + \text{FeatScore}(\text{rp}) + \text{FeatScore}(\text{gt}) \\
= \logprob(\text{role}) + \logprob(\text{dir}|\text{role}) + \logprob(\text{rp}|\text{role}, \text{dir}, h_{\text{gt}}) + \logprob(\text{gt}|h_{\text{gt}}, \text{role}, \text{dir}, \text{rp})
\]
Search Strategy

- **Branch-and-bound search**
  - Sum of feature logprobs is cost to be minimized
- **Case splitting**
  - First stage: split on var whose two most likely values have the greatest difference in likelihood
  - Second stage: split on var with greatest number of suspensions
- **Ordering**
  - Try values in order of greatest likelihood
Probabilistic Modeling

Issues

- Dynamic conditioning = finer-grained declarativeness
- Challenge: combinatorial explosion
- Alternatives:
  - Simulate with statically trained model
    - Marginalizing
    - Dynamic programming
    - Train a model for each decomposition
  - Approximate
    - Substitute most common value for unknown context feature
    - Lump conditioning contexts into equivalency classes
Dynamic smoothing (aka lazy learning)

- Ideal for accuracy, but
- Studied relatively little
- Slower
- We are in progress of researching
Sample sentence

<table>
<thead>
<tr>
<th>Token</th>
<th>Role</th>
<th>GT</th>
<th>AP</th>
<th>RP</th>
<th>Dir</th>
<th>ID</th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>subj</td>
<td>np</td>
<td>1</td>
<td>-1</td>
<td>-</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>flies</td>
<td>top</td>
<td>c</td>
<td>2</td>
<td>0</td>
<td>+</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>like</td>
<td>rma</td>
<td>o</td>
<td>3</td>
<td>-2</td>
<td>-</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>an</td>
<td>det</td>
<td>o</td>
<td>4</td>
<td>-1</td>
<td>-</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>arrow</td>
<td>ajt</td>
<td>np</td>
<td>5</td>
<td>1</td>
<td>+</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>.</td>
<td>rpunc</td>
<td>o</td>
<td>6</td>
<td>2</td>
<td>+</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
## Experimental Runs

<table>
<thead>
<tr>
<th>Given VS. To-Be-Determined</th>
<th>Search Tree Size</th>
<th>Prev. Search Tree Size</th>
<th>Total Search Tree Size</th>
<th>Depth</th>
<th>Num Sols</th>
<th>Best Sol</th>
<th>Num Vars</th>
<th>% Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>token, role, head VS. dir, rp, ap, gt</td>
<td>70</td>
<td>194</td>
<td>2376</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>24</td>
<td>100</td>
</tr>
<tr>
<td>token, role, ap VS. dir, rp, head, gt</td>
<td>71</td>
<td>259</td>
<td>2772</td>
<td>9</td>
<td>2</td>
<td>12</td>
<td>24</td>
<td>100</td>
</tr>
<tr>
<td>token, ap VS. role, dir, rp, head, gt</td>
<td>5485</td>
<td>NA</td>
<td>63756</td>
<td>16</td>
<td>2</td>
<td>17</td>
<td>30</td>
<td>97</td>
</tr>
</tbody>
</table>
Search Path to Best Solution: 1st Run (realization)

<table>
<thead>
<tr>
<th></th>
<th>Token</th>
<th>Rel</th>
<th>F</th>
<th>Vals</th>
<th>V</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>time</td>
<td>head</td>
<td>gt</td>
<td>c,np,o</td>
<td>c</td>
<td>0.99</td>
</tr>
<tr>
<td>2</td>
<td>an</td>
<td>head</td>
<td>gt</td>
<td>np,o</td>
<td>np</td>
<td>0.99</td>
</tr>
<tr>
<td>3</td>
<td>time</td>
<td>self</td>
<td>dir</td>
<td>-,+</td>
<td>-</td>
<td>0.93</td>
</tr>
<tr>
<td>4</td>
<td>time</td>
<td>self</td>
<td>gt</td>
<td>np,o</td>
<td>np</td>
<td>0.62</td>
</tr>
<tr>
<td>5</td>
<td>arrow</td>
<td>self</td>
<td>dir</td>
<td>+,-</td>
<td>+</td>
<td>0.60</td>
</tr>
<tr>
<td>6</td>
<td>arrow</td>
<td>self</td>
<td>rp</td>
<td>1,2</td>
<td>1</td>
<td>0.58</td>
</tr>
<tr>
<td>7</td>
<td>an</td>
<td>self</td>
<td>rp</td>
<td>1,2</td>
<td>1</td>
<td>0.22</td>
</tr>
</tbody>
</table>
### Search Path to Best Solution: 2nd Run (sort of parsing)

<table>
<thead>
<tr>
<th></th>
<th>Token</th>
<th>Rel</th>
<th>F</th>
<th>Vals</th>
<th>V</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>time</td>
<td>head</td>
<td>gt</td>
<td>c,np,o</td>
<td>c</td>
<td>0.99</td>
</tr>
<tr>
<td>2</td>
<td>time</td>
<td>self</td>
<td>gt</td>
<td>np,o</td>
<td>np</td>
<td>0.62</td>
</tr>
<tr>
<td>3</td>
<td>arrow</td>
<td>head</td>
<td>gt</td>
<td>o,c</td>
<td>c</td>
<td>0.55</td>
</tr>
<tr>
<td>4</td>
<td>an</td>
<td>head</td>
<td>gt</td>
<td>np,o</td>
<td>np</td>
<td>0.99</td>
</tr>
<tr>
<td>5</td>
<td>arrow</td>
<td>self</td>
<td>rp</td>
<td>1,2</td>
<td>1</td>
<td>0.58</td>
</tr>
<tr>
<td>6</td>
<td>.</td>
<td>head</td>
<td>gt</td>
<td>o,np,c</td>
<td>not o</td>
<td>0.43</td>
</tr>
<tr>
<td>7</td>
<td>.</td>
<td>head</td>
<td>gt</td>
<td>c,np</td>
<td>c</td>
<td>0.93</td>
</tr>
<tr>
<td>8</td>
<td>like</td>
<td>head</td>
<td>gt</td>
<td>o,np</td>
<td>np</td>
<td>0.43</td>
</tr>
</tbody>
</table>
### Search Path to Best Solution: 3rd Run (parsing)

<table>
<thead>
<tr>
<th>Token</th>
<th>Rel</th>
<th>F</th>
<th>Vals</th>
<th>V</th>
<th>Diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>an</td>
<td>self</td>
<td>rol</td>
<td>det</td>
<td>0.99</td>
</tr>
<tr>
<td>2</td>
<td>flies</td>
<td>self</td>
<td>dir</td>
<td>+</td>
<td>0.82</td>
</tr>
<tr>
<td>3</td>
<td>time</td>
<td>self</td>
<td>rol</td>
<td>ajt</td>
<td>0.75</td>
</tr>
<tr>
<td>4</td>
<td>like</td>
<td>self</td>
<td>rol</td>
<td>rma</td>
<td>0.72</td>
</tr>
<tr>
<td>5</td>
<td>arrow</td>
<td>head</td>
<td>gt</td>
<td>c</td>
<td>0.55</td>
</tr>
<tr>
<td>6</td>
<td>an</td>
<td>head</td>
<td>gt</td>
<td>np</td>
<td>0.99</td>
</tr>
<tr>
<td>7</td>
<td>arrow</td>
<td>self</td>
<td>rp</td>
<td>1</td>
<td>0.64</td>
</tr>
<tr>
<td>8</td>
<td>an</td>
<td>head</td>
<td>rol</td>
<td>ajt</td>
<td>0.49</td>
</tr>
<tr>
<td>9</td>
<td>.</td>
<td>head</td>
<td>gt</td>
<td>Not o</td>
<td>0.43</td>
</tr>
<tr>
<td>10</td>
<td>.</td>
<td>head</td>
<td>gt</td>
<td>c</td>
<td>0.93</td>
</tr>
<tr>
<td>11</td>
<td>like</td>
<td>head</td>
<td>gt</td>
<td>np</td>
<td>0.43</td>
</tr>
</tbody>
</table>
A Happy Marriage of Approaches: Probabilistic and Symbolic

**Pros**
- Automatically learnable
- Scaleable, robust systems
- Graded judgments

**Cons**
- Sparse data/smoothing
- Difficult to distinguish infrequent from impossible

**Pros**
- Strong inferences (efficient)
- Linguistic generalizations

**Cons**
- Labor-intensive
- Combinatorial explosion when rarer events are allowed
Most closely related CP work

- Denys Duchier et al
- Claire Gardent et al.
- Tomasz Marciniak et al.
- Dan Roth et al.

- Active prop., but no statistics
- ILP
A declarative approach is probably the most appropriate way to address the challenges of high-quality realization for MT

A hybrid approach combining hard and soft constraints offers computational advantages

Need further research to address lazy learning and additional tractability issues