XMG, Tree Adjoining Grammars and Unification Based Semantics

Claire Gardent

LORIA
Campus Scientifique,
BP 239,
F-54 506 Vandœuvre-lès-Nancy, France

Nancy - 2007, June
Outline

Unification based semantics in TAG

Using XMG to enrich a TAG with a unification based semantics

Factorisation vs. Principles
Compositional Semantics

Each word is associated with a meaning representation.

Each syntactic rule is associated with a semantic rule specifying how the semantics of the daughter constituents combine to yield the semantics of the mother constituent.
Two ways to define a Compositional Semantics

Lambda based

- The semantic representation of a word is a lambda term
- Composition of meanings involves functional application and beta reduction
- E.g., Montague, LFG glue semantics, CCG Boxer module, TAG (Pogodalla)

Unification based

- The semantic representation of a word is a feature structure
- Composition by unification
- E.g., HPSG, UCG, TAG (Stone, Frank & van Genabith, Gardent & Kallmeyer)
Unification based Semantics in FTAG

1. Each elementary tree is associated with a formula $\phi$ representing its meaning where parameters (missing arguments) are represented by unification variables.

2. Elementary tree nodes are decorated with unification variables occurring in $\phi$.

3. The meaning of a derived tree is the union of the meanings associated with the elementary trees under the unifications made during processing.
Exemple : “John loves Mary”

\[ \text{name}(j,\text{john}) \quad l_0:\text{love}(X,Y) \quad \text{name}(m,\text{mary}) \]

\[ \Rightarrow l_0:\text{love}(j,m),\text{name}(j,\text{john}),\text{name}(m,\text{mary}) \]
Using XMG to integrate semantic information

Each elementary tree must be:

1. Associated with a semantic representation
   Using the **SEM** dimension

2. Enriched with semantic indices
   Using **FEATURE STRUCTURES**

3. Synchronised with this semantic representation (linking between tree and formulae indices)
   Using **INTERFACES**
Example: “John runs”

\[
S \\
\downarrow [idx=X] \quad \text{VP} \quad = \quad l_0: \text{Rel}(X)
\]

\[
S \\
\downarrow [idx=X_t] \quad \text{VP} \quad \land \quad l_0: \text{Rel}(X_s) \quad \land \quad \text{subjI}=X_t
\]

\[
\text{subjI}=X_t \quad \land \quad \text{arg1}=X_s
\]

\[
\text{SEM} \quad \land \quad \text{IF}
\]
Factorisation

FraG, our Tree Adjoining Grammar for French, contains 6 000 trees

Factorisation is important

Factorisation through inheritance, parametrisation, colours and the combined use of $\lor$ and $\land$

Factorisation through principles
Semantic Representations

8 representation schemas, 75 schema calls

Factorisation through the use of:

1. schematic formulae
2. inheritance hierarchy
3. $\land$ and $\lor$
Factorisation using schematic formulae

Schematic formulae

Intransitive Verb $\rightarrow$ L:P(E) ; L:Theta₁(E,X)

... instantiated by the lexicon:

*Dormir* $\rightarrow$ P = dormir et Theta₁ = agent
Factorisation using the inheritance hierarchy

Through the use of the highest levels of the Inheritance hierarchy (one representation inherited by several syntactic classes)

E.g., prepositions: 10 syntactic classes, 1 semantic representation
Semantic representations

Factorisation using $\land$ and $\lor$

Through **TAG verb families** and the combined use of $\land$ and $\lor$

E.g., n0Vn1: **130 trees, one semantic representation**

\[
n0Vn1 \rightarrow \text{binaryRel} \land
\quad (\ \text{dian0Vn1Active}
\lor \text{dian0Vn1Passive}
\lor \text{dian0Vn1dePassive}
\lor \text{dian0Vn1ShortPassive}
\lor \text{dian0Vn1ImpersonalPassive}
\lor \text{dian0Vn1middle}
\lor \text{dian0Vn1Reflexive} ) \tag{1}
\]
**Specifying Tree Indices**

Two types of indices depending on the node they label:

**Requirement node:** substitution or foot node

- Index correlates with a missing semantic argument
- Hence must be named i.e., associated with an interface constraint

**Resource node:** internal or root node

- Provides a missing argument
- Index named and often percolated from the anchor (head) upwards
Substitution nodes

One semantic class per grammatical function e.g.

- Class name: SubjectSem
- Export: xSem
- Tree description: xSem[idx=X]
- Interface: subjI = X

Diagram with S, N, VP nodes andsubjI = X_t
Substitution nodes

Each possible realisation of a grammatical function imports the semantic class defined for that function

<table>
<thead>
<tr>
<th>Class name</th>
<th>CanonicalSubject</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import</td>
<td>SubjectSem</td>
</tr>
<tr>
<td>Tree description</td>
<td>xSubj(mark=subst)[cat = n]</td>
</tr>
<tr>
<td></td>
<td>xSubj = xSem</td>
</tr>
</tbody>
</table>
Substitution nodes

13 grammatical functions $\Rightarrow$ 13 semantic classes

58 argument realisations $\Rightarrow$ 58 semantic enrichments
Foot nodes

Adjectives, Adverbs, Prepositions

- Same principle: semantic class imported by syntactic class
- The set of semantic classes used is more restricted (no grammatical functions)
- Import done high in the hierarchy and inherited by all children

Sentential arguments: treated like the other arguments
Root and internal nodes (Resource Nodes)

Root and internal nodes are labelled through feature percolation (from the leaf nodes)

Feature percolation is achieved using colours because the same tree fragment must be combined with many different trees (with different node names)
Feature percolation using colours

Percolation specified once and specified to various category combinations:

\[
\begin{align*}
\circ_x, l \\
\circ_x, l \land \\
\circ_x, l \\
S \\
\rightarrow \\
S \circ_x, l \\
\circ \circ_x, l
\end{align*}
\]

One percolation tree, 7 specialisations
Semantic indices

Feature percolation using colours

Specialised percolation used to enrich the various syntactic trees:

\[
\begin{align*}
S_{\circ}^{x,l} & \quad \wedge & \quad S_{\bullet} \quad \rightarrow & \quad S_{\bullet}^{x,l} \\
S_{\odot}^{x,l} & \quad \vee & \quad S_{\bullet} \quad \rightarrow & \quad S_{\bullet}^{x,l} \\
\end{align*}
\]
Interfaces

Shared variables are used to unify indices across the syntactic and the semantic dimension

\[ n0Vn1 \rightarrow \text{binaryRel}[arg0=E,arg1=X,arg2=Y] \land
\begin{align*}
&\text{dian0Vn1Active}[vbl=E,subjectI=X,objectI=Y] \\
&\lor \text{dian0Vn1Passive}[vbl=E,subjectI=Y,cagentI=X] \\
&\lor \text{dian0Vn1dePassive}[vbl=E,subjectI=Y,deobjectI=X] \\
&\lor \text{dian0Vn1ShortPassive}[vbl=E,subjectI=Y] \\
&\lor \text{dian0Vn1ImpersonalPassive}[vbl=E,objectI=Y] \\
&\lor \text{dian0Vn1middle}[vbl=E,subjectI=Y] \\
&\lor \text{dian0Vn1Reflexive}[vbl=E,subjectI=X,refll=X]
\end{align*}\]
Factorisation vs. Principles

Using the inheritance hierarchy and colours allow a reasonably economical specification of the semantic information.

An even more economical way would be to use principles (Gardent 2007)

4 principles + linking rules
Argument labelling principle

Each argument node is labelled with a distinct variable named with its grammatical function.
Anchor Projection Principle

Each anchor projects its index up to the root (modulo appropriate renaming).
Foot Projection Principle

In a modifier auxiliary tree, root and foot node share semantic index.
Controller/Controllee Principle

In a control verb tree, controller and controllee index are the same.
Linking rules

n0Vn1, active
subject = object

n0Vn1, active, impersonal subj
object

n0Vn1, passive
object = subject
deobj = subject

etc.

arg1 = arg2 =
Remark

- Labelling principles may vary depending on a particular semantic implementation

- However (Gardent 2007):
  - they seem limited in number
  - they seem partially reusable across semantic approaches
Conclusion

Integrating semantic information into a real size grammar is a complex task

- English CCG: roughly 1,200 categories i.e. lambda terms
- French TAG: roughly 6,000 elementary trees

Some means of (semi-)automating the task is needed to:

- reduce the risk of error (typos, etc.)
- facilitate extension and maintenance
- reduce development time

XMG provides some good means of minimising redundancies
Principles would be even better!
Thanks!