Hybrid Logic Dependency Semantics

What goes into the OpenCCG surface realiser?

Lecture 2
January 22, 2013

Brief recap - What is NLG?

How computer programs can be made to produce (high-quality) natural language text from
- computer-internal representations of information
- other texts

Brief recap - the NLG pipeline

Communicative Goal(s)

Document Planning

Document Plans

Microplanning

Sentence Plans

Surface Realisation

Surface Text

The first part of this course is concerned with the bottom part of the pipeline:

surface realisation

Using the (state-of-the-art) OpenCCG surface realiser

Today’s lecture:

What do OpenCCG sentence plans look like?

i.e. What are “hybrid logic dependency structures”?

Surface realisation with OpenCCG

Sentence Plan

OpenCCG realiser

Lexicon

Grammar

Surface Text
An OpenCCG sentence plan

Hybrid logic:
@x restaurant ^ @x <THEME> w ^ @y inexpensive ^
@y <THEME> w ^ @z attractive ^ @z <THEME> w ^
@w Giovanni's

XML:
<satop nomvar="X">
  <prop name="restaurant"/>
</satop>

<satop nomvar="X">
  <diamond mode="theme">
    <nomvar name="W"/>
  </diamond>
</satop>

English:
"Giovanni's is an attractive inexpensive restaurant."

Directed graphs

OpenCCG input representations are fundamentally directed graphs:

- **nodes** - "points"
- **edges** - "arrows" connecting two points

If there is an edge from node X to node Y, there is a dependency between entities X and Y
- i.e., Y is a dependent of X

Directed graphs - topological constraints?

OpenCCG directed graphs can be either **trees** or **non-trees**.

"Re-entrancy" and "multi-rootedness" are permitted

Directed graphs - topological constraints?

OpenCCG directed graphs can even be **non-connected** or **cyclic**
Labelled directed graphs

OpenCCG input representations are labelled directed graphs:

- **node labels** - different types of entity
- **edge labels** - different types of relation/dependency

Surface realisation with OpenCCG

Surface sentence plans

- Basicly a syntactic dependency structure.
- Possible realisations are highly constrained:
  - Giovanni's is an attractive inexpensive restaurant.
  - Giovanni's is an inexpensive attractive restaurant.

Deep sentence plans

- Many possible realisations:
  - Giovanni's is an attractive inexpensive restaurant.
  - Giovanni's is both inexpensive and attractive.
  - Giovanni's does cheap food and has attractive decor.
  - At Giovanni's, the food offers good value. Moreover, the decor is attractive.
Intermediate sentence plans

More realisations than surface, but fewer than deep:

- Giovanni's is an attractive inexpensive restaurant.
- Giovanni's is a restaurant that is cheap and attractive.
- Giovanni's, an attractive restaurant, serves inexpensive food.

Graphs and logic

Think about first order logic.

Formulas:

- $\exists x \ \exists y. \ x \neq y \ ^{\land} \ boy(x) \ ^{\land} \ girl(y) \ ^{\land} \ love(x,y) \ ^{\land} \ \neg love(y,x)$

Every formula describes a set of models:

- the set of models in which the formula is true

Models are graphs!

Node labels are unary predicates (properties).
Edge labels are binary predicates (relations).

Surface realisation with OpenCCG

Sentence plan:

- a labelled directed graph

How can we represent labelled directed graphs?
i.e., we need a graph description language (a logic)

Graphs and logic

Want to encode labelled directed graphs in a linear format

i.e., convert graph into a logic formula that describes just that graph

But which logical system shall we use to encode labelled directed graphs?

- first-order logic is way more expressive than we need
- modal logic is perfectly suited to describe graphs (Kripke structures) - hence used in OpenCCG
Describing directed graphs - modal propositional logic

The <> modal operator is used to signal a link between two nodes (i.e. a relation/dependency between two entities).

Another example

Describing labelled directed graphs - multimodal propositional logic

Instead of just one modal operator <>, there is a range of different multimodal operators, e.g., <PRICE>, <DECOR>

- denote different types of relation/dependency between two entities

Another example
Graphs that are not trees?

But: normal modal logic has no way of ensuring that it is the same entity which is the THEME in all three cases.

Also, no way of combining the three fragments into a single formula - conjunction won't do.

Hybrid multimodal logic - nominals

Nominals capture reentrancy, but not multi-rootedness.

Elementary predications

Every hybrid logic formula can be turned into an equivalent conjunction of elementary predications (EP)

Two kinds of EP:
1. Node label statements:
   - @x restaurant
   - node x is labelled “restaurant”
2. Edge statements:
   - @x <THEME> y
   - there is an edge labelled “theme” from node x to node y
Elementary predications

Hybrid logic formula:

$$\forall x \text{ restaurant} \land \forall y \text{ inexpensive} \land \forall z \text{ attractive}$$

Conjunction of EPs:

$$\forall x \text{ restaurant} \land \forall y \text{ inexpensive} \land \forall z \text{ attractive}$$

Another example

$$\text{be} \land (\text{SBJ Giovanni's}) \land (\text{PRED} (\text{restaurant} \land (\text{DET} a) \land (\text{MOD} \text{attractive}) \land (\text{MOD} \text{inexpensive}))))$$
**Elementary predications in XML**

**Node label statements:** @x attractive

```
<satop nomvar="X">
  <prop name="attractive"/>
</satop>
```

**Edge statements:** @x <THEME> y

```
<satop nomvar="X">
  <diamond mode="theme">
    <nomvar name="Y"/>
  </diamond>
</satop>
```

**What you need to know**

How to convert a labelled directed graph into a set of elementary predications of hybrid multimodal logic

How to convert a set of elementary predications of hybrid multimodal logic into a labelled directed graph

• Reading for Week 2:

**Learn more about hybrid logic**


