Sentence Realisation with OpenCCG

Lecture 4

February 1, 2013

Recap

- What have you learned so far?
  - hybrid logic dependency structures (HLDSs)
    - What goes into the OpenCCG surface realiser?
  - categorial grammars (CGs)
  - integrating HLDSs into CGs
- Today:
  - Combinatory Categorial Grammar
    - an extension of Categorial Gramar
  - How chart realisation works

So where are we?

- We've seen how to define a lexicon in CG
- We've learned about two important operators in CG, i.e., forward and backward application
- We've seen how to combine words both
  - Syntactically (derivations, unification), **and**
  - Semantically (set union of elementary predications)
- But, Combinatory Categorial Grammar gives us more expressive power

From CG to CCG

CCG is an "extension" of CG

CCG has more rules:
- forward and backward type raising
- forward and backward composition

Everything else remains the same
- in particular the HLDS representations.

Forward type raising

$$\frac{X}{Y/(Y\backslash X)}\ T$$

$$\dfrac{\dfrac{\text{John}}{NP}}{S/(S\backslash NP)}\ T$$

Type Raising

- CCG includes type-raising rules, which turn arguments into functions over functions over such arguments
- Forward type raising

$$\frac{X}{Y/(Y\backslash X)}\ T$$

- Example:

$$\dfrac{\dfrac{\text{John}}{NP}}{S/(S\backslash NP)}\ T$$

- Rules are order preserving. Here we turn an NP into a rightward looking function over leftward functions, preserving the linear order of constituents

Forward composition

$$\frac{X/Y \qquad Y/Z}{X/Z}\ \mathbf{\mathit{B}}$$

$$\dfrac{\dfrac{\text{John}}{S/(S\backslash NP)} \qquad \dfrac{\text{likes}}{(S\backslash NP)/NP}}{S/NP}\ \mathbf{\mathit{B}}$$

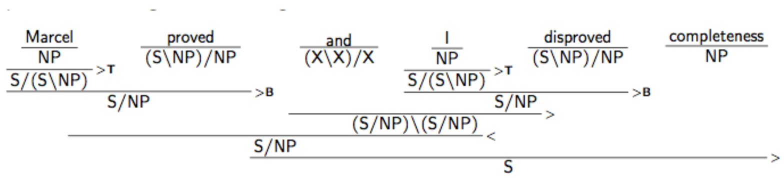
CCG is more flexible

CCG generates **more** sentences:

- object relative clauses –
  "a restaurant that [John likes]$_{S/NP}$"
- right node raising –
  "[John likes]$_{S/NP}$ but [Charles hates]$_{S/NP}$ Giovanni's"

Right Node Raising

Example:



CCG is more flexible

CCG allows one sentence to be derived in many ways

- reflecting different intonation patterns
- allowing incremental (i.e. left-branching) derivations from a right-branching lexicon

Multiple derivations

Q1: *I know what restaurant serves French food, but what restaurant serves Italian food?*

A1: Babbo serves Italian food.

NP S\NP/NP NP

S\NP

Q2: *I know what kind of food Pierre's serves, but what kind of food does Babbo serve?*

A2: Babbo serves Italian food.

NP S\NP/NP NP

T

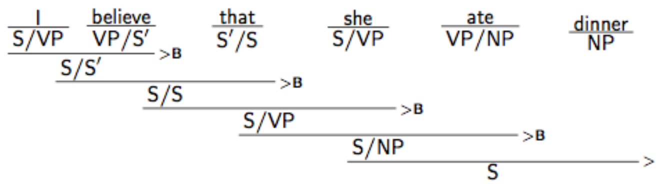
S/(S\NP)

B

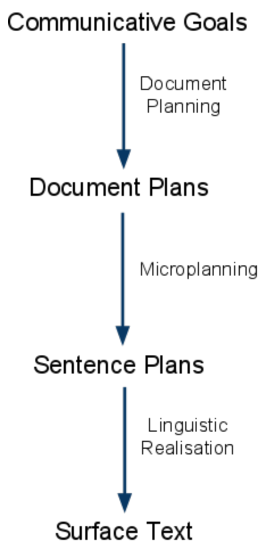
S/NP

CCC allows Incremental Processing

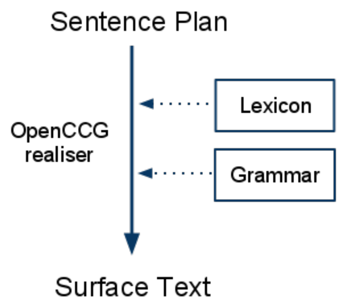
Can interpret from left to right



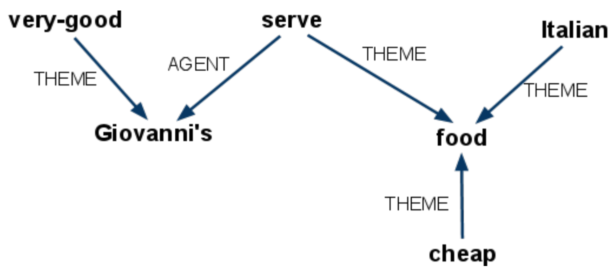
Modularity - the NLG pipeline



Linguistic realisation with OpenCCG

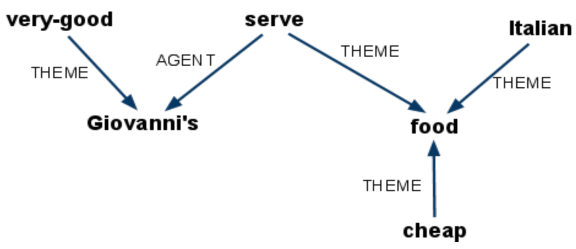


Sentence plans are labelled directed graphs



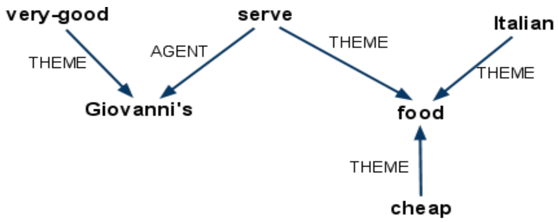
i.e. nodes, edges, labels

Labelled directed graphs can be represented as hybrid logic formulas




```
  @e (very-good ^ <THEME> (x ^ Giovanni's))
^ @f (serve ^ <AGENT> x ^ <THEME> (y ^ food))
^ @g (Italian ^ <THEME> y)
^ @h (cheap ^ <THEME> y)
```

Sentence plans are sets of elementary predications in hybrid logic




```
{@e very-good, @x Giovanni's, @f serve, @y food,
@g Italian, @h cheap, @e <THEME> x, @f <AGENT> x,
@f <THEME> y, @g <THEME> y, @h <THEME> y}
```

Why hybrid logic?

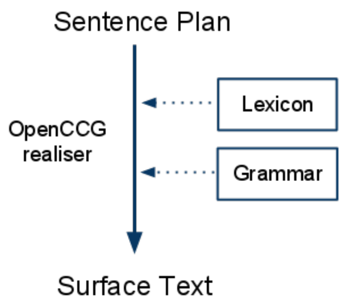
Ideal for representing labelled directed graphs
- extension of modal logic

Hybrid logic is well understood
- decidable fragment of first order logic

Hybrid logic allows a graph to be represented in two different, but equivalent, ways -
- hierarchically - good for viewing by humans
- flat - good for processing by computers

Flat represention are particularly useful for doing surface realisation.

Linguistic realisation with OpenCCG



Representing Syntax in XML

Atomic categories - S, NP, N, ...

```
<atomcat type="S"/>

<atomcat type="NP"/>

<atomcat type="N"/>
```

Complex categories - (N\N)/(S\NP)

```
<complexcat>
  <atomcat type="N"/>
  <slash dir="\"/>
  <atomcat type="N"/>
  <slash dir="/"/>
  <complexcat>
    <atomcat type="S"/>
    <slash dir="\"/>
    <atomcat type="NP"/>
  </complexcat
</complexcat>
```

Adding Semantics in XML

```
<satop nomvar="E">
  <prop name="very-good"/>
</satop>
```

```
@e very-good
```

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

```
@e <THEME> x
```

A grammar is a lexicon

Giovanni's :- NP

food :- N

Italian :- N/N

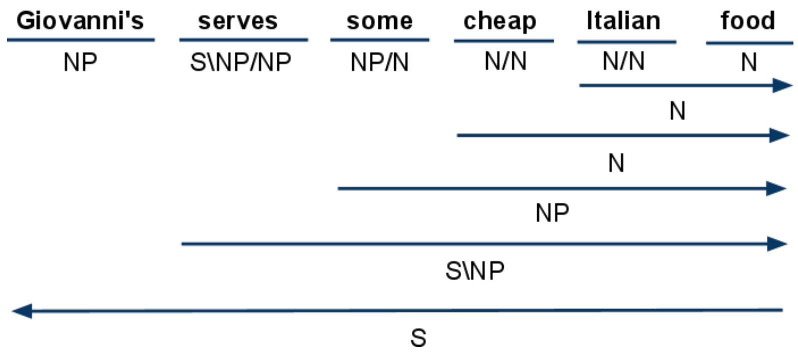
cheap :- N/N

rocks :- S\NP

serves :- S\NP/NP

some :- NP/N

Forward and backward application



Integrating HLDS - (1) Add nominals

Giovanni's :- NP_x

food :- N_x

Italian :- N_x/N_x

cheap :- N_x/N_x

rocks :- $S_e\backslash NP_x$

serves :- $S_e\backslash NP_x/NP_y$

some :- NP_x/N_x

Adding nominals in XML

```
<atomcat type="S">
  <fs>
    <feat attr="index">
      <lf>
        <nomvar name="x"/>
      </lf>
    </feat>
  </fs>
</atomcat>
```

Integrating HLDS - (2) Add EPs

Giovanni's :- NP_x : @x Giovanni's

food :- N_x : @x food

Italian :- N_x/N_x : @e Italian, @e <THEME> x

cheap :- N_x/N_x : @e cheap, @e <THEME> x

rocks :- $S_e\backslash NP_x$: @e very-good, @e <THEME> x

serves :- $S_e\backslash NP_x/NP_y$: @e serve, @e <AGENT> x, @e <THEME> y

some :- NP_x/N_x :

Semantic construction

Giovanni's	**serves**	**some**	**cheap**	**Italian**	**food**
NPx @x Giovanni's	Se\NPy/NPz @e serve @e <AGENT> y @e <THEME> z	NPw/Nw	Nv/Nv @f cheap @f <THEME> v	Nu/Nu @g Italian @g <THEME> u	Nt @t food

Nu
@g Italian
@g <THEME> u
@u food

Nv
@f cheap, @f <THEME> v, @g Italian
@g <THEME> v, @v food

Nw
@f cheap, @f <THEME> w, @g Italian
@g <THEME> w, @w food

Se\NPy
@e serve, @e <AGENT> y, @e <THEME> z, @f cheap,
@f <THEME> z, @g Italian, @g <THEME> z, @z food

Se : @e serve, @e <AGENT> y, @y Giovanni's, @e <THEME> z, @f cheap,
@f <THEME> z, @g Italian, @g <THEME> z, @z food

Type raising and composition

Giovanni's	serves	some	cheap	Italian	food
NPx	Se\NPy/NPz	NPw/Nw	Nv/Nv	Nu/Nu	Nt
@x Giovanni's	@e serve		@f cheap	@g Italian	@t food
	@e <AGENT> y		@f <THEME> v	@g <THEME> u	
	@e <THEME> z				

T: Sh/(Sh\NPx) @x Giovanni's

B: Sh/NPz @h serve, @h <AGENT> x, @x Giovanni's, @h <THEME> z

B: Sh/Nz : @h serve, @h <AGENT> x, @x Giovanni's, @h <THEME> z

B: Sh/Nz : @h serve, @h <AGENT> x, @x Giovanni's, @h <THEME> z, @f cheap, @f <THEME> z

B: Sh/Nz : @h serve, @h <AGENT> x, @x Giovanni's, @h <THEME> z, @f cheap, @f <THEME> z, @g Italian, @g <THEME> z

Sh : @h serve, @h <AGENT> x, @x Giovanni's, @h <THEME> z, @f cheap, @f <THEME> z, @g Italian, @g <THEME> z, @z food

Why CCG?

CCGs are lexicalised
- allows for efficient NLG

CCGs are powerful
- it is easy to generate sentences with unbounded dependencies (object relative clauses, right node raising)

CCGs are flexible
- we can simulate incremental processing
- easy to integrate with models of intonation for spoken language generation

CCGs have transparent semantics
- easy to integrate with HLDSs

Chart realisation

An algorithm for converting flat semantic representations into text, using a lexicalised grammar.

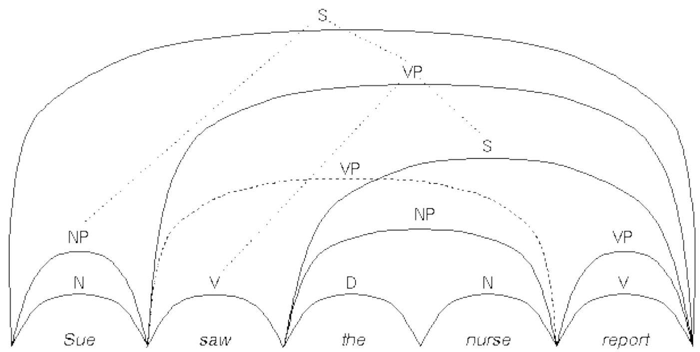
First proposed in Martin Kay (1996): "Chart generation" (ACL).

Adapted for OpenCCG by Michael White.

More efficient than other realisation algorithms
- semantic head-driven generation

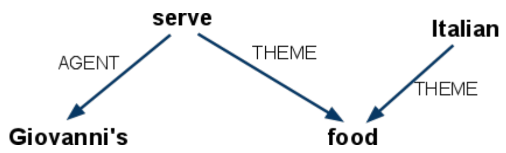
Inspired by chart parsing.

Chart parsing?



More efficient than normal top-down or bottom-up parsing
- keeps a record of what it has learned
- so doesn't have to keep repeating the same computations

Example - input sentence plan



1. @a serve
2. @b Italian
3. @c Giovanni's
4. @d food
5. @a <AGENT> c
6. @a <THEME> d
7. @b <THEME> d

Every sentence plan EP gets a unique index.

Example - lexicon

Giovanni's :- NP_x : @x Giovanni's
food :- N_x : @x food
Italian :- N_x/N_x : @y Italian, @y <THEME> x
cheap :- N_x/N_x : @y cheap, @y <THEME> x
rocks :- $S_x\backslash NP_y$: @x very-good, @x <THEME> y
serves :- $S_x\backslash NP_y/NP_z$: @x serve, @x <AGENT> y, @x <THEME> z
some :- NP_x/N_x :

Every entry has exactly one "indexing EP", represented by the underline.

Example - sentence plan + lexicon

1. @a serve
2. @b Italian
3. @c Giovanni's
4. @d food
5. @a <AGENT> c
6. @a <THEME> d
7. @b <THEME> d

Giovanni's :- NP_x : @x Giovanni's
food :- N_x : @x food
Italian :- N_x/N_x : @y Italian, @y <THEME> x
cheap :- N_x/N_x : @y cheap, @y <THEME> x
rocks :- $S_x\backslash NP_y$: @x very-good, @y <THEME> x
serves :- $S_x\backslash NP_y/NP_z$: @x serve, @x <AGENT> y, @x <THEME> z
some :- NP_x/N_x :

Step 1: add lexical edges

repeat for every EP φ in the sentence plan:
  repeat for every entry E in the lexicon:
    if E's indexing EP matches φ
    then add the relevant lexical edge
        to the chart.

1. @a serve
2. @b Italian
3. @c Giovanni's
4. @d food
5. @a <AGENT> c
6. @a <THEME> d
7. @b <THEME> d

Giovanni's :- NP_x : @x Giovanni's
food :- N_x : @x food
Italian :- N_x/N_x : @y Italian, @y <THEME> x
cheap :- N_x/N_x : @y cheap, @y <THEME> x
rocks :- $S_x\backslash NP_y$: @x very-good, @x <THEME> y
serves :- $S_x\backslash NP_y/NP_z$: @x serve, @x <AGENT> y, @x <THEME> z
some :- NP_x/N_x :

The chart (1)

serves
Sa\NPc/NPd
1,5,6
2-4,7

1. @a serve*
2. @b Italian
3. @c Giovanni's
4. @d food
5. @a <AGENT> c
6. @a <THEME> d
7. @b <THEME> d

Giovanni's :- NP_x : @x Giovanni's
food :- N_x : @x food
Italian :- N_x/N_x : @y Italian, @y <THEME> x
cheap :- N_x/N_x : @y cheap, @y <THEME> x
rocks :- $S_x\backslash NP_y$: @x very-good, @x <THEME> y
serves :- $S_x\backslash NP_y/NP_z$: @x serve, @x <AGENT> y, @x <THEME> z
some :- NP_x/N_x :

The chart (2)

serves	Italian
Sa\NPc/NPd	Nd/Nd
1,5,6	2,7
2-4,7	1,3-6

1. @a serve*
2. @b Italian*
3. @c Giovanni's
4. @d food
5. @a <AGENT> c
6. @a <THEME> d
7. @b <THEME> d

Giovanni's :- NP_x : @x Giovanni's
food :- N_x : @x food
Italian :- N_x/N_x : @y Italian, @y <THEME> x
cheap :- N_x/N_x : @y cheap, @y <THEME> x
rocks :- $S_x\backslash NP_y$: @x very-good, @x <THEME> y
serves :- $S_x\backslash NP_y/NP_z$: @x serve, @x <AGENT> y, @x <THEME> z
some :- NP_x/N_x :

The chart (3)

serves	Italian	Giovanni's	food
Sa\NPc/NPd	Nd/Nd	NPc	Nd
1,5,6	2,7	3	4
2-4,7	1,3-6	1,2,4-7	1-3,5-7

1. @a serve*
2. @b Italian*
3. @c Giovanni's*
4. @d food*
5. @a <AGENT> c
6. @a <THEME> d
7. @b <THEME> d

Giovanni's :- NP_x : @x Giovanni's
food :- N_x : @x food
Italian :- N_x/N_x : @y Italian, @y <THEME> x
cheap :- N_x/N_x : @y cheap, @y <THEME> x
rocks :- $S_x\backslash NP_y$: @x very-good, @x <THEME> y
serves :- $S_x\backslash NP_y/NP_z$: @x serve, @x <AGENT> y, @x <THEME> z
some :- NP_x/N_x :

1. @a serve*
2. @b Italian*
3. @c Giovanni's*
4. @d food*
5. @a <AGENT> c*
6. @a <THEME> d*
7. @b <THEME> d*

Giovanni's :- NP_x : @x Giovanni's
food :- N_x : @x food
Italian :- N_x/N_x : @y Italian, @y <THEME> x
cheap :- N_x/N_x : @y cheap, @y <THEME> x
rocks :- $S_x\backslash NP_y$: @x very-good, @x <THEME> y
serves :- $S_x\backslash NP_y/NP_z$: @x serve, @x <AGENT> y, @x <THEME> z
some :- NP_x/N_x :

Step 1: add lexical edges

repeat for every EP φ in the sentence plan:
  repeat for every entry E in the lexicon:
    if E's indexing EP matches φ
    then add the relevant lexical edge
        to the chart.

serves	Italian	Giovanni's	food
Sa\NPc/NPd	Nd/Nd	NPc	Nd
1,5,6	2,7	3	4
2-4,7	1,3-6	1,2,4-7	1-3,5-7

1. @a serve*
2. @b Italian*
3. @c Giovanni's*
4. @d food*
5. @a <AGENT> c*
6. @a <THEME> d*
7. @b <THEME> d*

Giovanni's :- NP_x : @x Giovanni's
food :- N_x : @x food
Italian :- N_x/N_x : @y Italian, @y <THEME> x
cheap :- N_x/N_x : @y cheap, @y <THEME> x
rocks :- $S_x\backslash NP_y$: @x very-good, @x <THEME> y
serves :- $S_x\backslash NP_y/NP_z$: @x serve, @x <AGENT> y, @x <THEME> z
some :- NP_x/N_x :

Step 1: add lexical edges (revised)

repeat for every EP φ in the sentence plan:
  repeat for every entry E in the lexicon:
    if E's indexing EP matches φ
    then add the relevant lexical edge
        to the chart.

repeat for every entry E in the lexicon:
  if E has no EPs
  then add the relevant lexical edge
        to the chart anyway.

The chart (4)

serves	Italian	Giovanni's	food	some
Sa\NPc/NPd	Nd/Nd	NPc	Nd	NPx/Nx
1,5,6	2,7	3	4	-
2-4,7	1,3-6	1,2,4-7	1-3,5-7	1-7

Step 2: apply CCG rules

repeat for every edge E on the chart:
  repeat for every edge F on the chart:
    if there is a CCG rule
      that can combine E and F
    and the EP in-sets of E and F are disjoint
    then add the relevant edge to the chart.

The chart (5)

serves Sa\NPc/NPd 1,5,6 2-4,7	Italian Nd/Nd 2,7 1,3-6	Giovanni's NPc 3 1,2,4-7	food Nd 4 1-3,5-7	some NPx/Nx - 1-7

The chart (6)

serves Sa\NPc/NPd 1,5,6 2-4,7	Italian Nd/Nd 2,7 1,3-6	Giovanni's NPc 3 1,2,4-7	food Nd 4 1-3,5-7	some NPx/Nx - 1-7
Italian food Nd 2,4,7 1,3,5,6				

The chart (7)

serves Sa\NPc/NPd 1,5,6 2-4,7	Italian Nd/Nd 2,7 1,3-6	Giovanni's NPc 3 1,2,4-7	food Nd 4 1-3,5-7	some NPx/Nx - 1-7
Italian food Nd 2,4,7 1,3,5,6				

The chart (8)

serves	Sa\NPc/NPd	1,5,6	2-4,7
Italian	Nd/Nd	2,7	1,3-6
Giovanni's	NPc	3	1,2,4-7
food	Nd	4	1-3,5-7
some	NPx/Nx	-	1-7
Italian food	Nd	2,4,7	1,3,5,6
some food	NPd	4	1-3,5-7

The chart (9)

serves	Sa\NPc/NPd	1,5,6	2-4,7
Italian	Nd/Nd	2,7	1,3-6
Giovanni's	NPc	3	1,2,4-7
food	Nd	4	1-3,5-7
some	NPx/Nx	-	1-7
Italian food	Nd	2,4,7	1,3,5,6
some food	NPd	4	1-3,5-7

The chart (10)

serves	Sa\NPc/NPd	1,5,6	2-4,7
Italian	Nd/Nd	2,7	1,3-6
Giovanni's	NPc	3	1,2,4-7
food	Nd	4	1-3,5-7
some	NPx/Nx	-	1-7
Italian food	Nd	2,4,7	1,3,5,6
some food	NPd	4	1-3,5-7
some Italian food	NPd	2,4,7	1,3,5,6

The chart (11)

serves	Sa\NPc/NPd	1,5,6	2-4,7
Italian	Nd/Nd	2,7	1,3-6
Giovanni's	NPc	3	1,2,4-7
food	Nd	4	1-3,5-7
some	NPx/Nx	-	1-7
Italian food	Nd	2,4,7	1,3,5,6
some food	NPd	4	1-3,5-7
some Italian food	NPd	2,4,7	1,3,5,6

The chart (12)

serves
Sa\NPc/NPd
1,5,6
2-4,7

Italian
Nd/Nd
2,7
1,3-6

Giovanni's
NPc
3
1,2,4-7

food
Nd
4
1-3,5-7

some
NPx/Nx
-
1-7

Italian food
Nd
2,4,7
1,3,5,6

some food
NPd
4
1-3,5-7

some Italian food
NPd
2,4,7
1,3,5,6

serves some food
Sa\NPc
1,4,5,6
2,3,7

The chart (13)

serves
Sa\NPc/NPd
1,5,6
2-4,7

Italian
Nd/Nd
2,7
1,3-6

Giovanni's
NPc
3
1,2,4-7

food
Nd
4
1-3,5-7

some
NPx/Nx
-
1-7

Italian food
Nd
2,4,7
1,3,5,6

some food
NPd
4
1-3,5-7

some Italian food
NPd
2,4,7
1,3,5,6

serves some food
Sa\NPc
1,4,5,6
2,3,7

serves some Italian food
Sa\NPc
1,2-7
3

The chart (14)

serves
Sa\NPc/NPd
1,5,6
2-4,7

Italian
Nd/Nd
2,7
1,3-6

Giovanni's
NPc
3
1,2,4-7

food
Nd
4
1-3,5-7

some
NPx/Nx
-
1-7

Italian food
Nd
2,4,7
1,3,5,6

some food
NPd
4
1-3,5-7

some Italian food
NPd
2,4,7
1,3,5,6

serves some food
Sa\NPc
1,4,5,6
2,3,7

serves some Italian food
Sa\NPc
1,2-7
3

Giovanni's serves some food
Sa
1,3-6
2,7

The chart (15)

serves
Sa\NPc/NPd
1,5,6
2-4,7

Italian
Nd/Nd
2,7
1,3-6

Giovanni's
NPc
3
1,2,4-7

food
Nd
4
1-3,5-7

some
NPx/Nx
-
1-7

Italian food
Nd
2,4,7
1,3,5,6

some food
NPd
4
1-3,5-7

some Italian food
NPd
2,4,7
1,3,5,6

serves some food
Sa\NPc
1,4,5,6
2,3,7

serves some Italian food
Sa\NPc
1,2-7
3

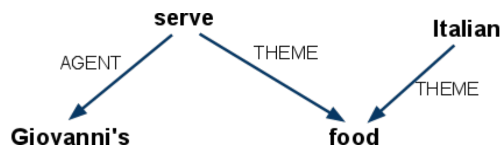
Giovanni's serves some food
Sa
1,3-6
2,7

Giovanni's serves some Italian food
Sa
1-7
-

The chart (16)

serves Sa\NPc/NPd 1,5,6 2-4,7	Italian Nd/Nd 2,7 1,3-6	Giovanni's NPc 3 1,2,4-7	food Nd 4 1-3,5-7	some NPx/Nx - 1-7
Italian food Nd 2,4,7 1,3,5,6	some food NPd 4 1-3,5-7	some Italian food NPd 2,4,7 1,3,5,6	serves some food Sa\NPc 1,4,5,6 2,3,7	
serves some Italian food Sa\NPc 1,2-7 3	Giovanni's serves some food Sa 1,3-6 2,7			
Giovanni's serves some Italian food Sa 1-7 -				

Result!



"Giovanni's serves some Italian food"

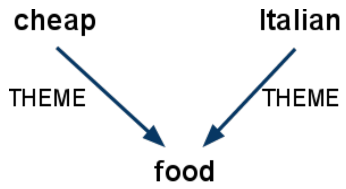
Chart realisation algorithm

1. repeat for every EP φ in the sentence plan:
   repeat for every entry E in the lexicon:
   if E's indexing EP matches φ
   then add the relevant lexical edge to the chart.

2. repeat for every entry E in the lexicon:
   if E has no EPs
   then add the relevant lexical edge to the chart.

3. repeat for every edge E on the chart:
   repeat for every edge F on the chart:
   if there is a CCG rule that can combine E and F
   and the EP in-sets of E and F are disjoint
   then add the relevant edge to the chart.

Another example



1. @a cheap
2. @b Italian
3. @c food
4. @a <THEME> c
5. @b <THEME> c

Giovanni's :- NP_x : <u>@x Giovanni's</u>
food :- N_x : <u>@x food</u>
Italian :- N_x/N_x : <u>@y Italian</u>, @y <THEME> x
cheap :- N_x/N_x : <u>@y cheap</u>, @y <THEME> x
rocks :- $S_x\backslash NP_y$: <u>@x very-good</u>, @x <THEME> y
serves :- $S_x\backslash NP_y/NP_z$: <u>@x serve</u>, @x <AGENT> y, @x <THEME> z
some :- NP_x/N_x :

The chart - lexical edges added

cheap
Nc/Nc
1,4
2,3,5

Italian
Nc/Nc
2,5
1,3,4

food
Nc
3
1,2,4,5

some
NPx/Nx
-
1-5

The chart - CCG rules applied

cheap
Nc/Nc
1,4
2,3,5

Italian
Nc/Nc
2,5
1,3,4

food
Nc
3
1,2,4,5

some
NPx/Nx
-
1-5

cheap food
Nc
1,3,4
2,5

Italian food
Nc
2,3,5
1,4

Italian cheap food
Nc
1-5
-

some food
NPc
3
1,2,4,5

some cheap food
NPc
1,3,4
2,5

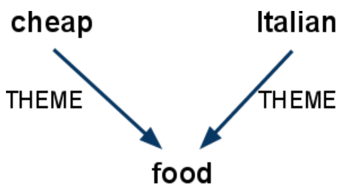
some Italian food
NPc
2,3,5
1,4

some Italian cheap food
NPc
1-5
-

cheap Italian food
Nc
1-5
-

some cheap Italian food
NPc
1-5
-

Result



"Italian cheap food"
"some Italian cheap food"
"cheap Italian food"
"some cheap Italian food"

One for the road

Giovanni's :- NP_x : <u>@x Giovanni's</u>
great :- A_x : <u>@x very-good</u>
rocks :- $S_x\backslash NP_y$: <u>@x very-good</u>, @x <THEME> y
is :- $S_x\backslash NP_y/A_x$: <u>@x <THEME> y</u>

What you need to know

Convert a labelled directed graph into a set of hybrid logic elementary predications, and vice versa.

Given a CCG lexicon, show how sentence S can be derived

- including semantic representations

Given a CCG lexicon, show how labelled directed graph G can be realised, using the chart realisation algorithm.