CS 533
Natural Language Processing
Lecture 6 – March 10, 2003

Matthew Stone

Department of Computer Science
Center for Cognitive Science
Rutgers University

Syntactic Representation Outline

Syntactic rules
Constituency
X-bar Theory
Substitution
Modification
Movement, traces, slashes

Syntactic representation

Further reading
Beatrice Santorini
An introduction to syntactic theory
http://www.ling.upenn.edu/~beatrice/150/toc-long.html
Chapters 1 – 5.

Syntactic rules

So far we've been describing individual utterances one-at-a-time.

Cats detest peas.
Children eat tomatoes.
Cheetahs chase gazelles.

Syntactic rules

But clearly there are more sentences than we can describe individually.

Sentence template: Ns V Ns.
Vocabulary size: N: 1000, V: 100
Number of sentences: 100 million.

Syntactic rules

Anyway, we can understand sentences we've never heard before.

Peas detest cats.
Tomatoes eat children.
Gazelles chase cheetahs.

Syntactic rules

At the same time, we know that other potential sentences are not part of English.

*Cats black detest peas green.
*Adopt which cat did your friend?
*Over there is the who I went to the party with guy.

In linguistics * labels ungrammatical sentences.

Syntactic rules

Our goal is to come up with representations and algorithms that characterize the sentences in a natural language, and what they mean.

Syntactic rules must be formulated in linguistically meaningful ways

To make a question in English:

Take the main auxiliary and move it to the beginning of the sentence (swap it with the subject of the sentence).

The girl is tall ->
Is the girl tall?
The red pig can stand on the house ->
Can the red pig stand on the house?

Syntactic rules must be formulated in linguistically meaningful ways

Impossible rule:

Take the ***first*** auxiliary and move it to the beginning of the sentence.

The boy who was holding the plate is crying
Was the boy who holding the plate is crying?
Is the boy who was holding the plate crying?

Linguistically meaningful rules must refer to

Syntactic constituency –
the natural ways words group together in sentences.

Syntactic relationships –
the kinds of meaningful combinations that link words together in sentences.

Constituency

What groups of words go together.

Evidence: ***substitution***

A constituent is any syntactic unit; a single word is the smallest possible such unit; if a single word can substitute for a string, that's evidence that the word and the string are constituents of the same category.

Constituency and substitution

The little boy fed the cat. ->
He fed the cat.

Black cats detest green peas. ->
They detest them.

Constituency

Constituency is a judgment about the intended analysis of particular utterance.

The little boy from next door fed the cat without a tail. ->
*He from next door fed her without a tail.
-> He fed her.

The little boy is sometimes a noun phrase, but not always.

Other kinds of constituents

There: prepositional phrases.

Put it on the table. -> Put it there.
Put it over on the table. -> Put it over there.
Put it over on the table. -> Put it there.

Put it over on the table that's by the door ->
*Put it there that's by the door.

Other kinds of constituents

So: adjective phrases.

I am very proud and Linda is so too.
I am very proud of the results and Linda is so too.

*I am very proud of the program and Linda is so of the documentation.

Other kinds of constituents

So or *it*: full sentences
I know that they're invited. -> I know it.
I imagine that they're invited -> I think so.

Other kinds of constituents

do so: verb phrases
She will give you a back rub ->
She will do so.

The two boys could order tuna salad sandwiches ->
The two boys could do so.

Representing constituenthood

Tree diagrams – the graphic structure of a tree is a statement about how the speaker intends to group together the words into constituents (mentally).

Syntactic constituents are represented graphically as nodes in the tree.

Representing constituenthood

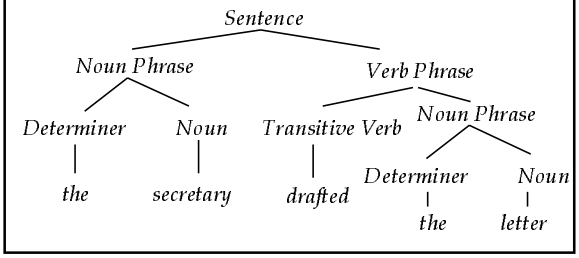
The secretary drafted the letter.

Need:

sentence: the secretary drafted the letter.
noun phrase: the secretary, the letter
verb phrase: drafted the letter.

Representing constituenthood

The secretary drafted the letter.



Other constituent tests

Can something function as a sentence fragment in response to a question?

What do you like?

The cats.
Cats with long, fluffy tails.
The cats with long, fluffy tails.

Other constituent tests

Can you construct an *it*-cleft *focusing* a particular string of words?

Ordinary cats detest the smell of citrus fruits.
It is ordinary cats that detest the smell of citrus fruits.
It is the smell of citrus fruits that ordinary cats detest.

Caveats about constituency

Tests so far deal with *phrasal* constituents (which stand on their own).

Single words are *lexical* constituents (you can switch words, etc.), this can be different.

Caveats about constituency

Two kinds of verb phrase: ***finite*** and ***nonfinite***.

Finite verb phrases carry tense, modality.

She *gave you a backrub.*

Nonfinite verb phrases

She will *give you a backrub.*

Only nonfinite verb phrases move, etc.

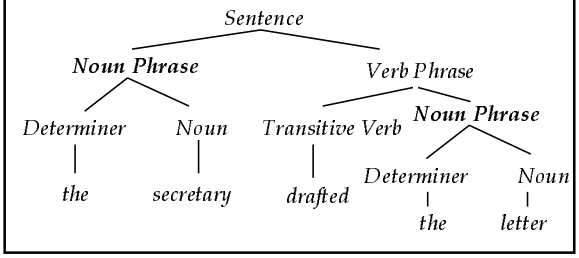
Constituenthood and rules

We've seen a close connection between ***substitution*** and ***constituency***.

Syntactic rules should allow constituents to substitute, where possible.

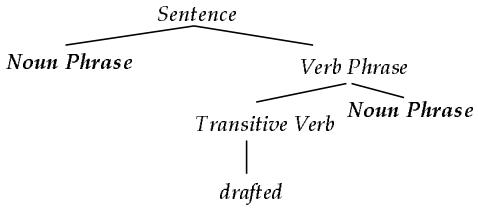
Constituenthood and rules

Substitution sites in bold.



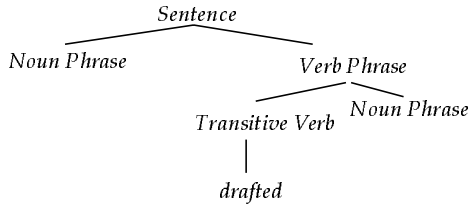
Constituenthood and rules

Substitution sites in bold.



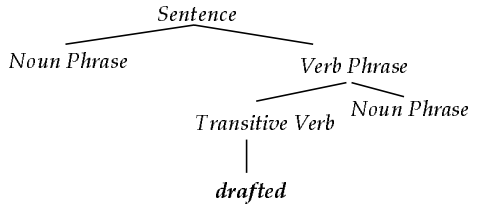
Constituenthood and rules

This kind of structure is called an ***elementary tree***.



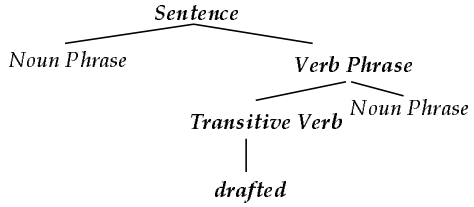
Talking about trees

Anchor or ***head***.



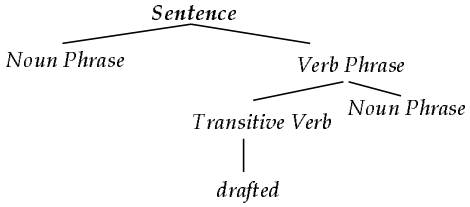
Talking about trees

Spine.



Talking about trees

Root.



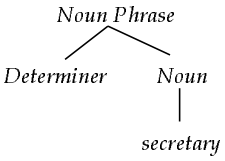
Constituenthood and rules

We can define grammatical operations that combine together elementary trees.

Substitution: unify the ***root*** of one tree with a ***substitution site*** in another tree (with the same label).

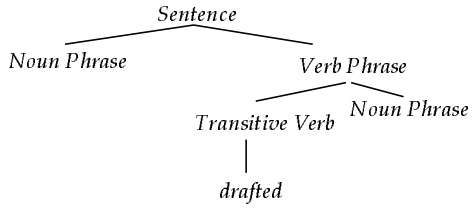
Illustrating substitution

One elementary tree



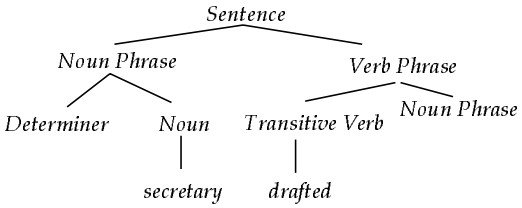
Illustrating substitution

Another elementary tree

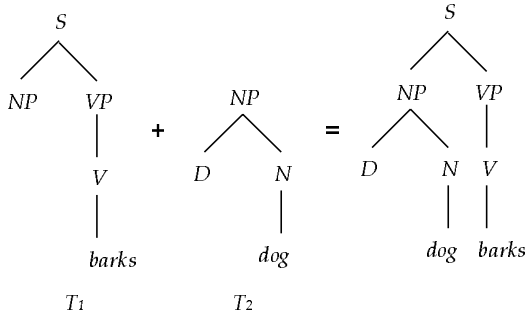


Illustrating substitution

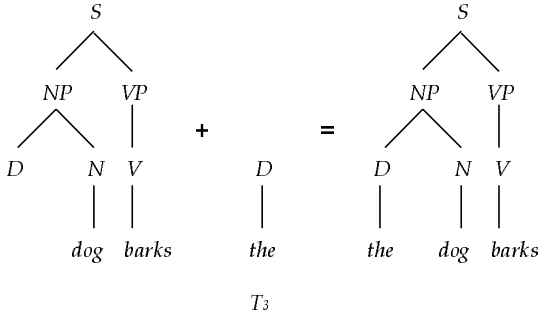
Unify root of one tree with substitution site in another.



Tree Substitution Grammar

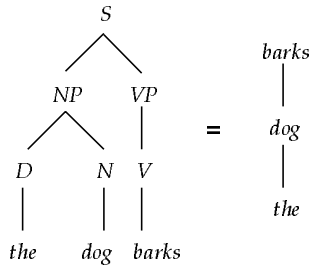


Tree Substitution Grammar



Dependency

Sentences built from words by operations



Dependency formalisms

Tree Substitution Grammar

barks: T1
-substitution at NP node
dog: T2
-substitution at D node
the: T3

Modification operations

There are lots of modifiers – modifying a constituent does not change its category.

The children ate the pizza.
The children ate the pizza with gusto.

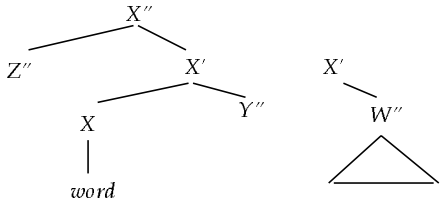
Modification operations

You can add modifiers indefinitely.

We would drink lemonade.
We would drink lemonade in the summer.
We would drink lemonade in the summer on the porch.
We would drink lemonade in the summer on the porch with friends.

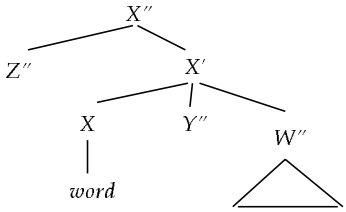
Possible modifications

Sister-adjunction



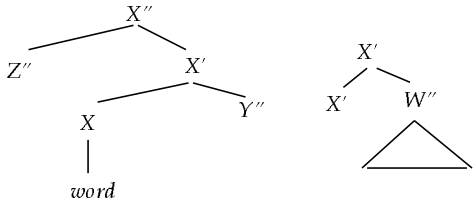
Possible modifications

Sister-adjunction



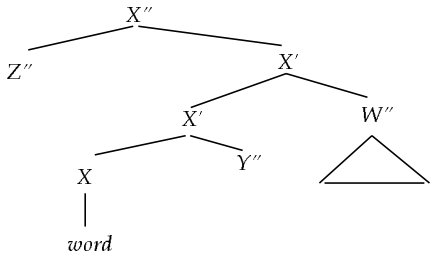
Possible modifications

General adjunction



Possible modifications

General adjunction



Modification issues

General adjunction:

- more syntactically-motivated

The children ate the pizza with gusto.
The parents did so also.
The parents did so reluctantly.

Modification issues

Sister-adjunction

- easier to implement
- context-free

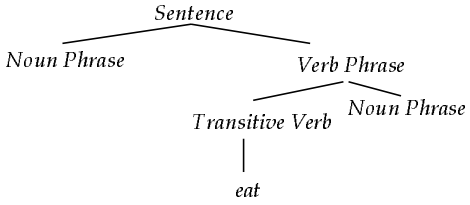
X-Bar theory

You need to generalize over elementary trees with similar structure.

(XTAG has thousands of trees for English!)

X-Bar theory

Start from here:



X-Bar Theory

This handles the sentence

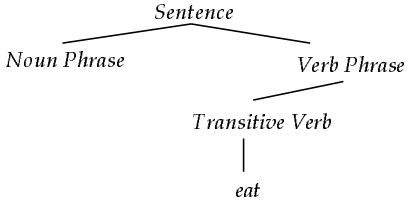
The children ate the pizza.

But what about

The children ate.

X-Bar theory

Keep the structure in common:



X-Bar theory

The head of the tree determines the category of projections along the spine.

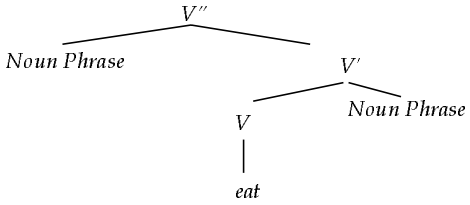
V -> V' -> V''

Complements come it at one bar level

Specifiers come in at two bar levels

X-Bar theory

Example



Parallels across categories

Predicate-argument structure

The army destroyed the city.
the army's destruction of the city.

The city was destroyed (by the army)
the city's destruction (by the army)

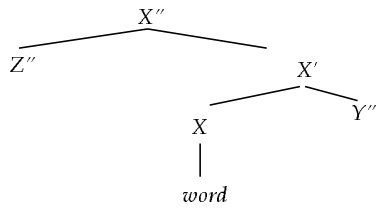
Parallels across categories

Modification

She gives money to the organization on a regular basis.
She regularly gives money to the organization on a regular basis.
her gifts of money to the organization on a regular basis
her regular gifts of money to the organization

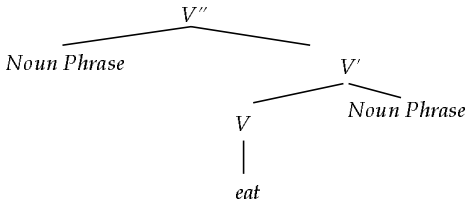
General X-Bar schema

Common underlying pattern for English



Movement

Elementary trees localize semantic and syntactic dependencies within a single constituent.



Movement

In some constructions, though, dependent elements appear outside this constituent.

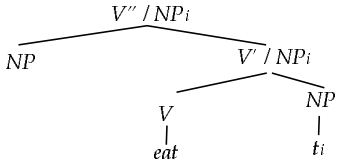
Contrast:

Bill knows Mary said the children ***ate pizza***.
Bill knows ***what*** Mary said they ***ate***.

Syntactic/semantic representations need to localize dependencies ***and*** encode word order.

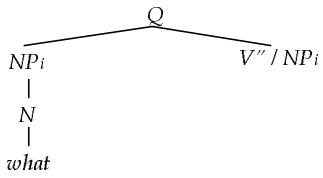
Movement – standard analysis

Trees with *gaps* (marked *t*) that show dependency but indicate extraction, and ***slashes*** (categories X/Y) that remember what's missing.



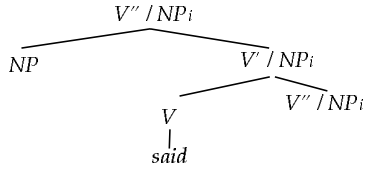
Movement – standard analysis

Filler trees that require slash categories, and eliminate them:

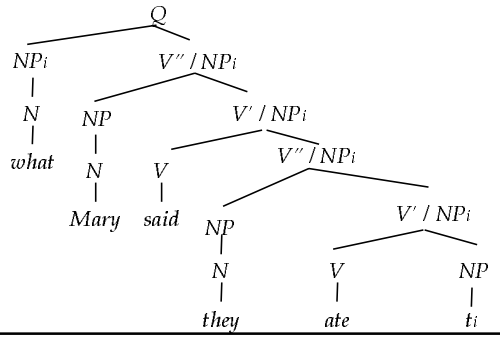


Movement – standard analysis

Gap threading trees that take slashes and pass them up.



Movement – putting it all together



Prolog Programming as Knowledge Representation

- Start by understanding the world.
  - Determine the ***objects*** that you need to consider; create Prolog terms for them.
  - Determine the ***relationships*** that you need to know about those objects; create Prolog predicates for them.
  - Develop a precise conceptualization that takes a stand on meaningful choices.
- State the facts.

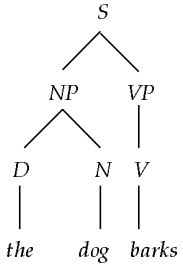
Case study: linguistic structure

- Objects:
  - Categories

```
s
np
d
n
vp
v
```

  - Words

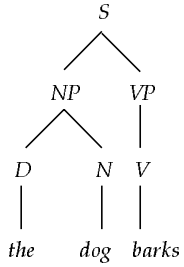
```
the
dog
barks
```



Linguistic data structures

- Lexical nodes:

```
leaf(the)
leaf(dog)
leaf(barks)
```

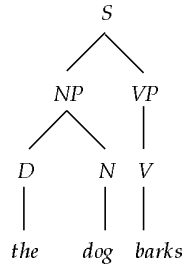


Linguistic data structures

- Internal nodes:

```
node(d,[leaf(the)])
node(n,[leaf(dog)])
node(v,[leaf(barks)])
```

```
node(np,
     [node(d,[leaf(the)]),
      node(n,[leaf(dog)])])
```



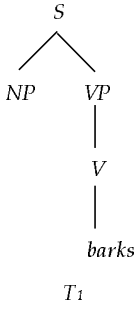
Linguistic data structures

```
node(s,
     [node(np,
           [node(d,[leaf(the)]),
            node(n,[leaf(dog)])]),
      node(vp,
           [node(v,[leaf(barks)])])])
```

Computing with Trees

- New object
  Substitution site or ***gap***
  with specific category

```
gap(np)
```



T_1

Computing with Trees

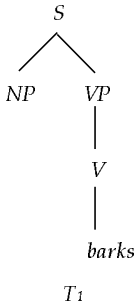
- Sample structure

```
node(s,
     [gap(np),
      node(vp,
           [node(v,
                 [leaf(barks)])
           ])
     ])
```

Computing with Trees

- New object
  Substitution site or ***gap***
  with specific category

```
gap(np)
```



T_1

Computing with Trees

- Sample structure

```
node(s,
     [gap(np),
      node(vp,
           [node(v,
                 [leaf(barks)])
           ])
     ])
```

Defining Substitution

New idea: two *mutually recursive* procedures
- Each represents the base case for the other.

For substitution, one procedure each for
- Whole nodes
- Lists of children

Defining Substitution

Overall goal:
- Define `subst(Tree,Subtree,Result)`
- Replace one `gap(C)` node in the passed `Tree` with a new `node(C,L)` `Subtree` to obtain `Result`

Basic basic case
- The passed `Tree` is itself `gap(C)`

Defining Substitution

Clause:

```
subst(gap(C),node(C,L),node(C,L)).
```

Otherwise, if `Tree` is a `node` replace one of its children.

```
subst(node(X,K), S, node(X,R)) :-
  subst_l(K, S, R).
```

Defining Substitution

Substitute somewhere in the list.

Base case: substitute in the head.

```
subst_l([Head|Tail],Subst,[R|Tail]):-
  subst(Head,Subst,R).
```

Defining Substitution

Recursive case: keep the head and substitute elsewhere.

```
subst_l([Head|Tail],Subst,[Head|R]):-
  subst_l(Tail,Subst,R).
```

Fancy illustration

```
? subst(node(s,[gap(np),
                node(vp,[node(v,[leaf(saw)]),
                         gap(np)])]),
        node(np,[leaf(chris)]),
        T).
```

Step 1 : Break down S

```
? subst(node(s,[gap(np),
                node(vp,[node(v,[leaf(saw)]),
                         gap(np)])]),
        node(np,[leaf(chris)]),
        T).
=>
? subst_l([gap(np),
           node(vp,[node(v,[leaf(saw)]),
                    gap(np)])],
          node(np,[leaf(chris)]),
          R).
```

Step 2, choice 1 – base case

```
? subst_l([gap(np),
           node(vp,[node(v,[leaf(saw)]),
                    gap(np)])],
          node(np,[leaf(chris)]),
          R).
=>
? subst(gap(np), node(np,[leaf(chris)]),
                 node(np,[leaf(chris)])).
=>
R = [node(np,[leaf(chris)]),
     node(vp,[node(v,[leaf(saw)]),gap(np)])]
```

Step 2, choice 1 – base case

```
R = [node(np,[leaf(chris)]),
     node(vp,[node(v,[leaf(saw)]),gap(np)])]
=>
T = node(s,
         [node(np,[leaf(chris)]),
          node(vp,[node(v,[leaf(saw)]),gap(np)])])
```

Step 2, choice 2– recurse

```
? subst_l([gap(np),
           node(vp,[node(v,[leaf(saw)]),
                    gap(np)])],
          node(np,[leaf(chris)]),
          R).
=>
? subst_l([node(vp,[node(v,[leaf(saw)]),
                    gap(np)])],
          node(np,[leaf(chris)]),
          R')
=>
R = [gap(np) | R']
```

Step 2, 2 – now base case

```
? subst_l([node(vp,[node(v,[leaf(saw)]),
                    gap(np)])],
          node(np,[leaf(chris)]),
          R').
=>
? subst(node(vp,[node(v,[leaf(saw)]),
                 gap(np)])],
          node(np,[leaf(chris)]),
          R'')
=>
R' = [ R'']
```

Step 2, 2 – recurse again

```
? subst(node(vp,[node(v,[leaf(saw)]),
                 gap(np)])],
          node(np,[leaf(chris)]),
          R'')
=>
R'' = node(vp,[node(v,[leaf(saw)]),
               node(np,[leaf(chris)])])
=>
R = [gap(np),
     node(vp,[node(v,[leaf(saw)]),
              node(np,[leaf(chris)])])]
```

Finally, then

```
R = [gap(np),
     node(vp,[node(v,[leaf(saw)]),
              node(np,[leaf(chris)])])]
=>
T = node(s,
         [gap(np),
          node(vp,[node(v,[leaf(saw)]),
                   node(np,[leaf(chris)])])])
```