Discontinuous Constituency

Parsing
with
Discontinuous Phrase Structure Grammar

- What is DPSG?
  - The Problem: Discontinuous Constituents
    - A Short History of DPSG
  - The Solution: DPSG
    - Definitions
- Parsing with DPSG
  - Active Chart Parser for DPSG
    - Parser Constituents
    - Basic Parsing Algorithm
    - Extension for Probabilistic Parsing
    - Evaluation
  - Future Work and Discussion

What Is DPSG?

A Short History of DPSG

Remember?

„Ich glaub, das mer d' Chind am Hans s' Huus lönd hälfe aastriiche."

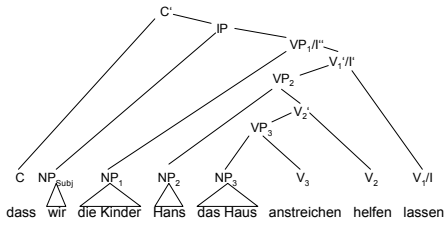
Phrase structure?

No problem in standard German:

„Ich glaube, dass wir die Kinder Hans das Haus anstreichen helfen lassen."

A Short History of DPSG

The phrase structure (PS) tree for standard German



A Short History of DPSG

But ordinary PS rules do not allow for discontinuous structures as needed in Swiss German...

What shall we do?

Solutions?

- Insertion of empty constituents
- Move-
- Traces and Indices
- etc.

Not very nice!

A Short History of DPSG

What do we need?

 A formalism that copes with discontinuous elements:

Discontinuous Phrase Structure Grammar (DPSG)

What do we get?

 Trees with crossing branches!

DPSG

Linguistic evidence:

- Many languages featuring discontinuous constituency:

  „*Was* hast du *für ein* Auto gekauft?“ vs.
  „*Was für ein* Auto hast du gekauft?“

- Every sentence can be made discontinuous by inserting metacomments:

  „John of course talked about politics.“

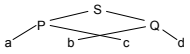
DPSG

- allows for better generalisations of syntactic structures
- is not a PS formalism of its own!
- works as „add-on“ to existing PSGs
- is thus particularly suitable for use in generation and parsing

DPSG

Discontinuous Trees (Discotrees)

Example:
S(P(a, [b], c), Q(b, [c], d))



Definitions:

- A discontinuous tree is a subdiscotree T in which every node except the root is immediately dominated by some other node in T.
- Subdiscotree: Subdiscotrees consist of a top node x and a sequence of daughter nodes immediately dominated by x, possibly interrupted by lists of nodes not dominated by x. These nodes are called *internal context* or *context daughters* of x.
- Immediate dominance: Every node in a subdiscotree immediately dominates all its real daughters, but not its context daughters.

DPSG

More definitions:

- Immediate dominance: Every node in a subdiscotree immediately dominates all its real daughters, but not its context daughters.
- Linear precedence (<):
  1. For two leaves x_i and x_j, $x_i < x_j$ if and only if $i < j$.
  2. For two arbitrary nodes x and y, $x < y$ if and only if
     a) the leftmost daughter of x precedes the leftmost daughter of y and
     b) there exists a node z to the right of Lm(x) that is either token-identical to or to the left of Lm(y), such that x does not dominate z.
- Adjacency (+): Two nodes x and y in a subdiscotree are adjacent (x + y), if and only if $x < y$ and there is no node z such that $x < z < y$.
- Adjacency sequence: A sequence ⟨a, b,..., n⟩ is an adjacency sequence if and only if
  1. every pair ⟨i, j⟩ in the sequence is either an adjacency pair or is connected by a sequence of adjacency pairs;
  2. the elements in the sequence do not share any constituents.

DPSG

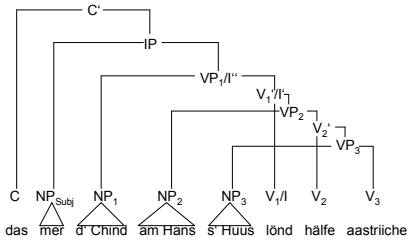
Finally:

A DPSG is a quadruple $\langle V_N, V_T, S, R \rangle$ where

- V_N is a finite set of nonterminal symbols;
- V_T is a finite set of terminal symbols; V denotes $V_N \cup V_T$;
- $S \in V_N$ is the distinguished start symbol of the grammar;
- R is a finite set of productions (rules) $X \rightarrow Y^{1,c1}Y^{2,c2}\ldots Y^{k,ck}$, where $X \in V_N$ and neither the the first nor the last right-hand side symbol may be marked as internal context.

DPSG

So...



DPSG

An example from the NEGRA corpus...

DPSG

Extracting DPSG Rules from a Corpus:

```
function extract_rules( discotree dt )
    for all nonterminal nodes n in dt do
        output(n);
        child=lm(n);
        while child is not n's last real daughter in left to right order do
            output(child);
            next_child ← sister(child);
            for all terminal nodes t such that lmd(child) < t < lmd(next_child)
                    in left to right order do
                if ¬(nDt) then
                    ctxt_dghtr ← highest node c such that cDt ∧ ¬(cDn)
                                 ∧ the yield of c is contained in
                                 between lmd(n) and rmd(n);
                    if ctxt_dghtr not already output for this rule then
                        output([ctxt_dghtr]);
            child ← next_child;
        output(child);
```

Parsing with DPSG

Parsing with DPSG

DPSG and Prolog

- DisLog: A Prolog library for reasoning in disjunctive deductive databases
  - a DPSG parser has been implemented for DisLog

Parsing with DPSG

- An Active Chart Parser for DPSG (Plaehn 1999)
  - Extensions for Parsing with DPSG
  - Parser Components
  - Basic Parsing Algorithm
  - Extension for Probabilistic Parsing
  - Evaluation
- Future Work

Active Chart Parser for DPSG

- Extensions for parsing with DPSG:
  - handling of context daughters: internal context of rules needs double processing, once as context and once an real daughter.
    → algorithm needs to keep track of which daughters already have a real mother.
  - combination of active and inactive edges gets more complicated: verification that the resulting edge forms an Adjacency Sequence.
    → „Adjacency-Sequence Test“

Active Chart Parser for DPSG

Parser Components:

- Categories
- Rules
- Edges
  - (start, end, rule, dot_position, covers, ctxt_covers, *children*)
  - [2, 5, NP → DET NOUN [VERB] • RELC, 00110000, 00001000]
- Agenda
- Chart

Active Chart Parser for DPSG

Basic parsing algorithm:

```
for i = 0 to n – 1 do
      edge ← new edge [i, i+1, t → •, 0^i, 1 0^(n-i-1), 0^n];
      agenda.add( edge );

while not agenda.is_empty() do
      edge ← agenda.get_next();
      chart.add(edge);

if chart.success(goal) then
      output successful parse(s);
else
      no parse for this sentence has been found;
```

Active Chart Parser for DPSG

Parser procedures:

- adding edges to the chart
- merging edges
- checking whether edges can be combined
- combining edges
- Adjacency-Sequence Test
- lookahead

→ The parser also works for normal CFGs, if presented with a non-discontinuous (context-free) sentence!

Probabilistic Parsing with DPSG

A Probabilistic DPSG (PDPSG) is a quintuple (V$_N$, V$_T$, S, R, P), where

- V$_N$ is a finite set of nonterminal symbols;
- V$_T$ is a finite set of terminal symbols; V denotes V$_N$ ∪ V$_T$;
- S ∈ V$_N$ is the distinguished start symbol of the grammar;
- R is a finite set of productions (rules) X → Y^{1,c_1}Y^{2,c_2}...Y^{k,c_k}, where X ∈ V$_N$ and neither the the first nor the last right-hand side symbol may be marked as internal context;
- P is a function R → [0, 1] that assigns each rule a probability such that the sum of all probabilities is 1.

Probabilistic Parsing with DPSG

Different probabilistic approaches:

- Edge-Based Best-First Chart Parsing
- Maximum Entropy Models
- etc.

→ Every probabilistic CFG Parser can be extended to DPSG Parsing!

Evaluation

	PCFG	PDPSG
Precision	79.24%	77.75%
Recall	78.09%	76.81%
F-score	78.66%	77.28%
CPU time	0.53 secs	24.78 secs

Evaluation

Why???

- parsing with discotrees is much harder than parsing with context-free trees
  - PCFG parsing: time ∝ (sentence length)^2
  - PDPSG parsing: time exponential to sentence length!
- the parser has to process may more edges

Future Work

- Restricting discontinuity
- Approximation algorithms
- Richer information sources

Questions and Discussion

?