Order-Preserving DAG Grammars, Parsing Complexity, and Learning

Henrik Björklund

Joint work with
Frank Drewes and Petter Ericson (Part 1)
Johanna Björklund and Petter Ericson (Part 2)

Umeå University

Porto, May 26, 2017
AMR for "the boy thinks that the girl likes him"
Motivation: AMR

Properties of AMRs:

- Directed and acyclic
- Reentrancies (not trees)
- Any number of modifiers (i.e. no fixed rank)
- No formalized grammar
Hyperedge replacement grammars
Hyperedge replacement grammars

[Diagram showing a transformation from a structure labeled 'A' to a structure labeled 'B' and 'a'.]
Hyperedge replacement grammars

- Context-free graph grammar
- Replace hyperedges with (hyper-)graphs
- Parsing studied by e.g. Lautemann, recently by Chiang et.al.
Hyperedge replacement grammars

- Context-free graph grammar
- Replace hyperedges with (hyper-)graphs
- Parsing studied by e.g. Lautemann, recently by Chiang et.al.
- Parsing NP-complete in the non-uniform case
Uniform vs. non-uniform parsing

For database theoreticians: Think data complexity vs. combined complexity

For verification people: Think model complexity vs. combined complexity
Uniform vs. non-uniform parsing

For database theoreticians: Think data complexity vs. combined complexity

For verification people: Think model complexity vs. combined complexity

Consider a grammar where we only have rules of the following forms:
Uniform vs. non-uniform parsing
Uniform vs. non-uniform parsing
Uniform vs. non-uniform parsing
Uniform vs. non-uniform parsing
Uniform vs. non-uniform parsing

We have arrived at the **UNORDERED CFG MEMBERSHIP** problem.
Uniform vs. non-uniform parsing

We have arrived at the **UNORDERED CFG MEMBERSHIP** problem.

This problem is **NP-complete**.
Uniform vs. non-uniform parsing

We have arrived at the **UNORDERED CFG MEMBERSHIP** problem.

This problem is **NP-complete**.

If, however, we look at the membership problem for a **fixed** CFL, it is solvable in polynomial time.
Order-preserving DAG grammars

Graph parsing is hard.
Order-preserving DAG grammars

Graph parsing is hard.

To achieve uniform polynomial parsing, we need to heavily restrict the right-hand sides.
Order-preserving DAG grammars

Graph parsing is hard.

To achieve uniform polynomial parsing, we need to heavily restrict the right-hand sides.
Order-preserving DAG grammars

Either the rule is a clone rule, or

1. all external (marked) nodes are leaves
2. edges from the root are terminal
3. nodes have out-degree at most one
4. leaves with in-degree 1 are either external or connected to a terminal edge
5. every non-leaf node has a terminal path from the root

Every rule also has to preserve the order of external nodes
It is not enough to avoid NP-hardness, we also want to avoid GI-hardness when matching a part of a graph to a right-hand side.
Parsing for OPDGs

- Work leaf-to-root
- From graph: determine external nodes (including order) and subgraphs
- Compare to rule right-hand sides
- In case of cloning rule: set intersection!
Parsing for OPDGs
Parsing for OPDGs
Parsing for OPDGs
A more complicated graph
We develop an algorithm for learning OPDGs from a *Minimally Adequate Teacher* (Angluin).

The teacher can answer
- **equivalence queries** (Is this the correct grammar?)
- **membership queries** (Does this graph belong to the language of the grammar?)
Graph operations

We need graph composition operations that correspond to the rule types of the grammar.

- **Clone concatenation** composes two graphs in parallel.
- **$\alpha$-concatenation** composes a number of graphs by “hanging” them under a terminal edge.
Concatenation
Clone-concatenation

\[
\begin{bmatrix}
a \\
b \\
c \\
b \\
b
\end{bmatrix}
= 
\begin{bmatrix}
a \\
b \\
c \\
b \\
b
\end{bmatrix}
\]
\( \alpha \text{-Concatenation} \)

\[
\begin{pmatrix}
\begin{array}{c}
\mathbf{b} \\
\mathbf{b} \\
\mathbf{b} \\
\mathbf{c}
\end{array}
\end{pmatrix}
\]

\( (g_1, g_2, g_3) = \)
α-Concatenation

\[ \langle m \rangle = \bullet \quad \bullet \quad \circ \quad \bullet \]
$\alpha$-Concatenation
A Myhill-Nerode theorem

**Theorem.** A DAG language $L$ can be generated by an OPDG if and only if $\equiv_L$ has finite index. If $\equiv_L$ has finite index, there is a unique minimal unambiguous OPDG for $L$. 
Theorem. An OPDG $G$ can be learned from a MAT in time polynomial in $|G|$ and the combined sizes of the counterexamples provided by the teacher.
The end

Thank you for listening!