Graph Transformation for Computational Linguistics

Frank Drewes

CiE 2014

Outline

1. Introduction
2. Graph Transformation
3. Context-Free Graph Generation by Hyperedge Replacement
4. Some Properties of Hyperedge Replacement Grammars
5. Recent Work Aiming at Computational Linguistics and NLP
6. Some More Questions for Future Work

Introduction

Introduction

Traditional grammars and automata used in Computational Linguistics work on strings.

- Context-free grammars, linear context-free rewriting systems, multiple context-free grammars, ...

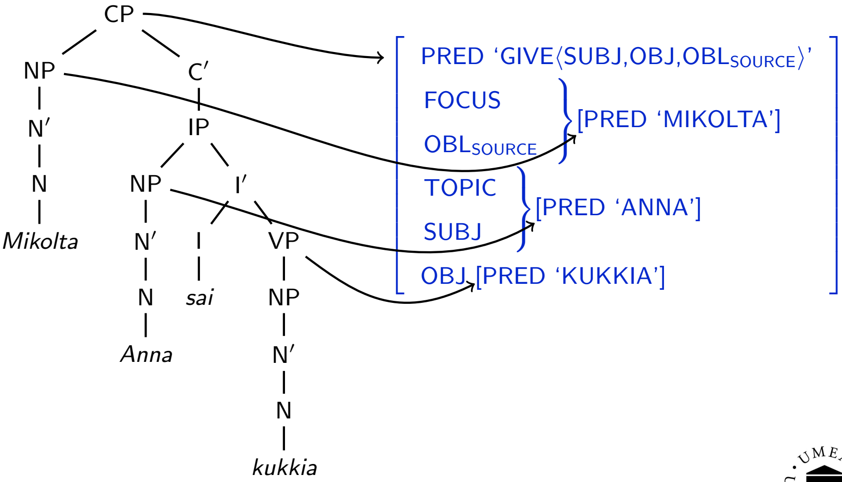
These formalisms were extended to trees to be able to handle sentence structure explicitly.

- Regular tree grammars, tree-adjoining grammars, context-free tree grammars, ...

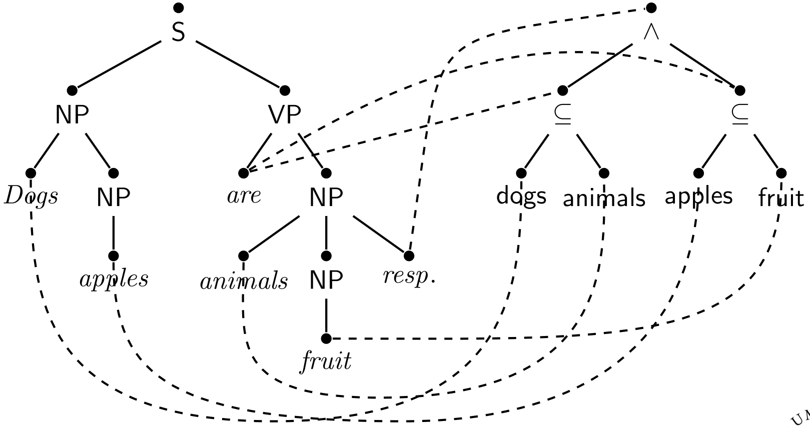
However, in many cases this is not enough.

We rather need to talk about graphs.

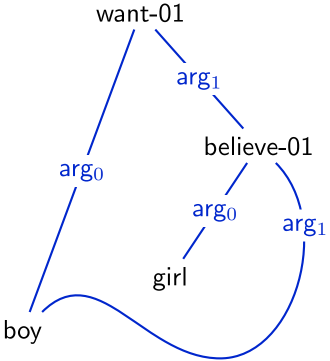
Example (LFG): "Mikolta Anna sai kukkia" [Dalrymple 2001]



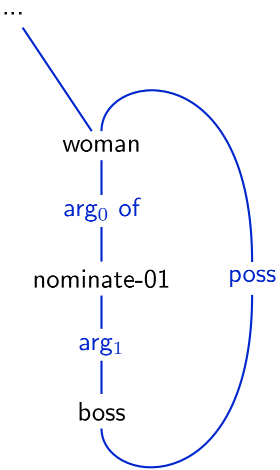
Example (Millstream system): "Dogs and apples are animals and fruit, respectively."



Example (Abstract Meaning Representation): "The boy wants the girl to believe him." [Banarescu et al. 2014]



Another AMR example: "...the woman who nominated her boss"



Many further examples can be found.

⇒ Computational Linguistics / NLP could benefit from suitable formalisms for generating and transforming graphs.

Such formalisms are provided by the theory of graph transformation that emerged around 1970.

This talk focusses in particular on hyperedge replacement grammars [Bauderon & Courcelle 1987], [Habel & Kreowski 1987].

Graph Transformation

General idea of graph transformation

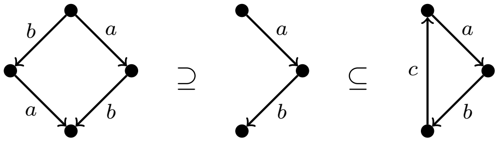
- Use rewriting rules $L \Rightarrow R$ that say "subgraph L can be replaced by R".
- Similar to string and term rewriting.
- However, how do you insert R in place of L?
- Different answers were given, e.g., connecting instructions and interface graphs (double-pushout approach, [Ehrig et al. 1973]).

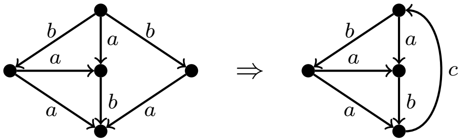
A few words about the double-pushout approach

- A rule is a triple $r = (L \supseteq I \subseteq R)$ of graphs.
- The middle graph I is the interface between L and R.
- To apply r to a graph G
  1. locate (a copy of) L in G,
  2. remove $L - I$, and
  3. add $R - I$.

I is the part where old and new items overlap.

Example





Context-Free Graph Generation by Hyperedge Replacement

Hyperedge Replacement

What would be a suitable context-free way of generating graphs?

Idea:

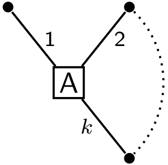
- A derivation step should replace an atomic item.
- Two possibilities:
  - replace nodes $\rightarrow$ node replacement grammars
  - replace edges $\rightarrow$ edge replacement grammars

$\downarrow$

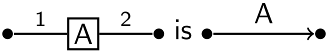
hyperedge replacement grammars

Hypergraphs

A hyperedge with label A of rank k:



Ordinary directed edges are included:



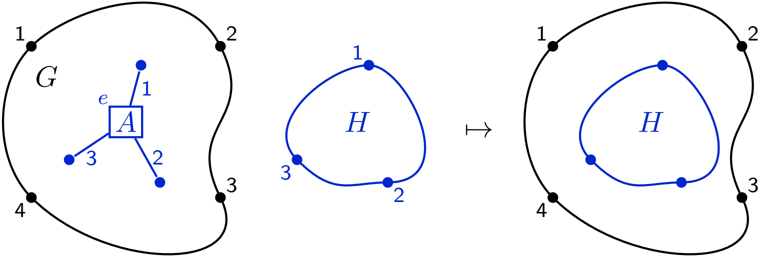
A hypergraph of rank k consists of

- nodes (usually unlabelled),
- hyperedges, and
- a sequence of k distinguished nodes called ports.

The Replacement Operation

A hyperedge e of rank k in a hypergraph G can be replaced by a hypergraph H of rank k:

1. build $G - e$ (remove e from G)
2. add H disjointly to $G - e$
3. fuse the k nodes to which e was attached with the ports of H.



Hyperedge Replacement Grammars

A hyperedge (HR) replacement grammar $\mathcal{G}$ has rules $A \Rightarrow H$, where

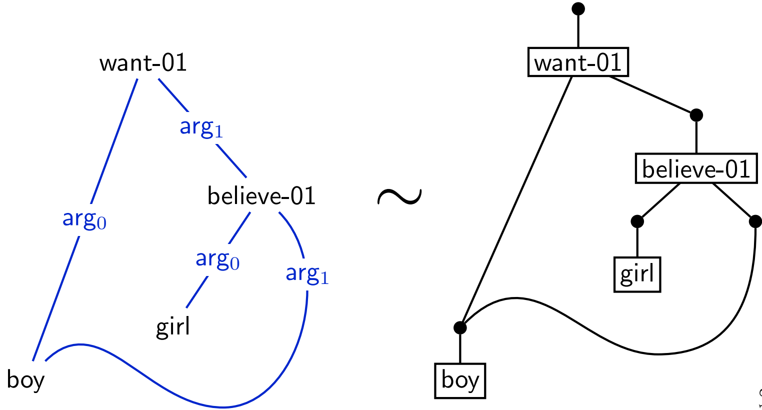
- A is a nonterminal hyperedge label of rank $k \in \mathbb{N}$ and
- H is a hypergraph of the same rank k.

Starting from a start graph, rules are applied until no nonterminal hyperedges are left.

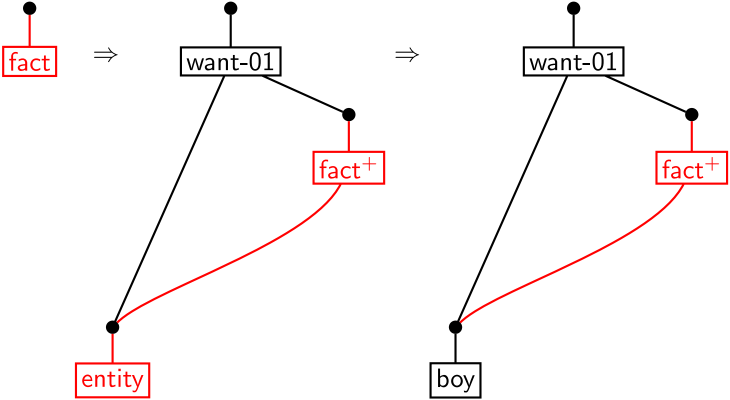
This yields the language $\mathcal{L}(\mathcal{G})$ generated by $\mathcal{G}$.

Example: “The boy wants the girl to believe him.”
What a derivation could possibly look like.

To be generated:



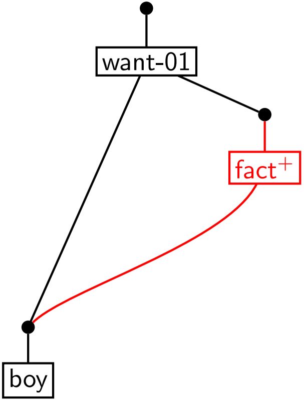
Example: "The boy wants the girl to believe him."
What a derivation could possibly look like.



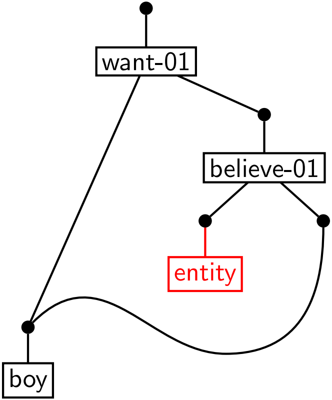
Hyperedge Replacement

Example: “The boy wants the girl to believe him.”
What a derivation could possibly look like.

⇒



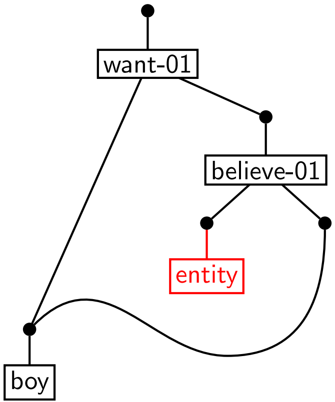
⇒



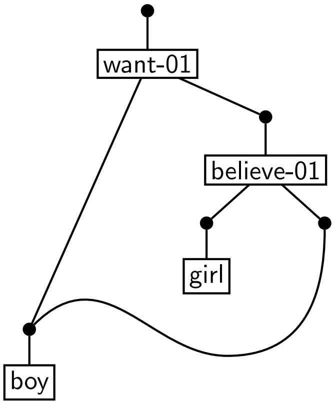
Hyperedge Replacement

Example: “The boy wants the girl to believe him.”
What a derivation could possibly look like.

⇒



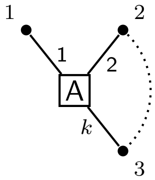
⇒



Some Properties of
Hyperedge Replacement Grammars

Some Properties of HRGs

Hyperedge replacement is context-free

For a nonterminal symbol A of rank k let $A^\bullet =$ 

Context-Freeness Lemma

There is a derivation $A^\bullet \Rightarrow^{n+1} G$ if and only if there exist

- a rule $A \Rightarrow H$ and nonterminals $e_1, \ldots, e_k$ with labels $A_1, \ldots, A_k$ in H and
- derivations $A_i^\bullet \Rightarrow^{n_i} G_i$

such that $G = [e_1/G_1, \ldots, e_k/G_k]$ and $n = \sum_{i=1}^{k} n_i$.

Some Properties of HRGs

Hyperedge replacement and mild context-sensitivity

String languages generated by hyperedge replacement

The string languages generated by HRGs are the same as those generated by

- deterministic tree-walking transducers,
- multiple context-free grammars,
- multi-component TAGs,
- linear context-free rewriting systems,
- ...

[Engelfriet & Heyker 1991]

Some Properties of HRGs

- A normal form similar to Chomsky normal-form exists
- Many properties of HR languages are decidable (work by Courcelle, Habel et al., Lengauer & Wanke about inductive/compatible/finite properties)
- Nice connections between HR and monadic second order logic on graphs (though not as perfect as for strings and trees)
- Parikh images are semi-linear (quite obviously, using Parikh's result)

Some Properties of HRGs

About parsing

- HR can generate NP-complete graph languages [Lange & Welzl 1987]
- Polynomial parsing possible in certain cases [Lautemann 1990], [Vogler 1991], [D. 1993], [Chiang et al. 2013]
- Major root of complexity is the large number of ways in which graphs can be decomposed
- Proof by Lange & Welzl very versatile ⇒ rules out many special cases one might otherwise hope to be easier

Recent Work Aiming at Computational Linguistics and NLP

Recent Work

Synchronous hyperedge replacement grammars

- [Jones et al. 2012] propose synchronous HRGs for semantics-based machine translation.
- Graphs represent the meaning of a sentence (Abstract Meaning Represenations).
- Synchronous HRGs are defined “as one would expect”.
- These are further extended by probabilities and then trained.

Note: Even though nobody seems to have mentioned it anywhere, synchronous HRGs are simply HRGs.

Recent Work

Lautemann's Parsing Algorithm Revisited

- [Chiang et al. 2013] refine Lautemann's parsing algorithm.
- Aim: Improve its efficiency in practice, so that it can be used in NLP settings.
- Main novelty: Use tree decompositions in order to reduce the search space.

Recent Work

Bolinas

- Bolinas is a software toolkit implementing synchronous HRGs, parsing, training, and other relevant algorithms.
- Developed at USC/ISI, see [Andreas et al. 2013].
- Implemented in Python.

Recent Work

Readers for incremental sentence analysis

- [Bensch et al. 2014] use graph transformation to turn a sentence into a graph that represents its analysis.
- Each word w is associated with a set $\Lambda(w)$ of rules.
- A derivation

$$G_0 \underset{\Lambda(w_1)}{\Rightarrow} G_1 \underset{\Lambda(w_1)}{\Rightarrow} \cdots \underset{\Lambda(w_n)}{\Rightarrow} G_n$$

  is a reading of $w_1 \cdots w_n$ that yields the analysis G_n.
- Soundness w.r.t. a so-called Millstream system turns out to be decidable.

A prototype implementation by F. Rozenberg under the supervision of H. Jürgensen (Western University, Ontario) is underway.

Some More Questions for Future Work

Some More Questions

Efficient parsing for cases that occur in Computational Linguistics?

- A typical type of structures that occurs in Computational Linguistics are directed acyclic graphs (DAGs).
- In general, HR languages of dags can be NP-complete (easy adaptation of the proof by Lange & Welzl).

⇒ Aim: identify cases not covered by known parsing algorithms, in which parsing is nevertheless "easy".

Some More Questions

From strings to graphs

- The input to an NLP task is often a sentence.
- An analysis of the sentence is a graph (cf. LFG, HPSG, AMR).

How do we get from one to the other?

Readers are a first attempt, but further techniques must be explored.

Some More Questions

Implementations?

- Bolinas implements HRGs and its algorithms from scratch.
- General purpose implementations of graph transformation have existed for quite a while.
- In particular, GrGen [Geiß et al. 2006] (now GrGen.net [Jakumeit et al. 2010]) is very efficient and powerful.

⇒ Question: Can systems such as GrGen.net be of interest as a basis for implementations?

Note: F. Rozenberg's implementation of readers uses GrGen.net.

Some More Questions

There is a lot to do;
most questions have not even been asked yet.

Please join if you are interested.

Thank you!

References I

J. Andreas, D. Bauer, K.M. Hermann, B. Jones, K. Knight, and D. Chiang.
A primer on graph processing with bolinas, 2013.

L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider.
Abstract meaning representation for sembanking.
In *Proc. 7th Linguistic Annotation Workshop, ACL 2013 Workshop*, 2013.

M. Bauderon and B. Courcelle.
Graph expressions and graph rewriting.
Mathematical Systems Theory, 20:83–127, 1987.

S. Bensch, F. Drewes, H. Jürgensen, and B. van der Merwe.
Graph transformation for incremental natural language analysis.
Theoretical Computer Science, 531:1–25, 2014.

D. Chiang, J. Andreas, D. Bauer, K.M. Hermann, B. Jones, and K. Knight.
Parsing graphs with hyperedge replacement grammars.
In *Proc. 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013), Volume 1: Long Papers*, LNCS, pages 924–932, 2013.

References II

M. Dalrymple.
Lexical Functional Grammar, volume 34 of *Syntax and Semantics*.
2001.

F. Drewes.
Recognising k–connected hypergraphs in cubic time.
Theoretical Computer Science, 109:83–122, 1993.

J. Engelfriet and L. Heyker.
The string generating power of context–free hypergraph grammars.
Journal of Computer and System Sciences, 43:328–360, 1991.

H. Ehrig, M. Pfender, and H.-J. Schneider.
Graph-grammars: An algebraic approach.
In *Proc. 14th Annual Symposium on Switching and Automata Theory (SWAT'08)*, pages 167–180, 1973.

R. Geiß, G. Veit Batz, D. Grund, S. Hack, and A.M. Szalkowski.
GrGen: A fast SPO-based graph rewriting tool.
In Corradini, Ehrig, Montanari, Ribeiro, and Rozenberg, editors, *Proc. 3rd Intl. Conf. on Graph Transformation (ICGT'06)*, volume 4178 of LNCS, pages 383–397, 2006.

References III

A. Habel and H.-J. Kreowski.
May we introduce to you: Hyperedge replacement.
In *Proceedings of the Third Intl. Workshop on Graph Grammars and Their Application to Computer Science*, volume 291 of LNCS, pages 15–26, 1987.

B. Jones, J. Andreas, D. Bauer, K.M. Hermann, and K. Knight.
Semantics-based machine translation with hyperedge replacement grammars.
In M. Kay and C. Boitet, editors, *Proc. 24th Intl. Conf. on Computational Linguistics (COLING 2012): Technical Papers*, pages 1359–1376, 2012.

E. Jakumeit, S. Buchwald, and M. Kroll.
GrGen.NET.
Intl. Journal on Software Tools for Technology Transfer, 12:263–271, 2010.

C. Lautemann.
The complexity of graph languages generated by hyperedge replacement.
Acta Informatica, 27:399–421, 1990.

K.-J. Lange and E. Welzl.
String grammars with disconnecting or a basic root of the difficulty in graph grammar parsing.
Discrete Applied Mathematics, 16:17–30, 1987.

References IV

W. Vogler.
Recognizing edge replacement graph languages in cubic time.
In H. Ehrig, H.-J. Kreowski, and G. Rozenberg, editors, *Proceedings of the Fourth Intl. Workshop on Graph Grammars and Their Application to Computer Science*, volume 532 of *LNCS*, pages 676–687. 1991.