Report from Dagstuhl Seminar 15122

Formal Models of Graph Transformation in Natural Language Processing

Edited by

Frank Drewes[1], Kevin Knight[2], and Marco Kuhlmann[3]

1 **Umeå University, SE,** drewes@cs.umu.se
2 **University of Southern California, US,** knight@isi.edu
3 **Linköping University, SE,** marco.kuhlmann@liu.se

Abstract

In natural language processing (NLP) there is an increasing interest in formal models for processing *graphs* rather than more restricted structures such as strings or trees. Such models of graph transformation have previously been studied and applied in various other areas of computer science, including formal language theory, term rewriting, theory and implementation of programming languages, concurrent processes, and software engineering. However, few researchers from NLP are familiar with this work, and at the same time, few researchers from the theory of graph transformation are aware of the specific desiderata, possibilities and challenges that one faces when applying the theory of graph transformation to NLP problems. The Dagstuhl Seminar 15122 "Formal Models of Graph Transformation in Natural Language Processing" brought researchers from the two areas together. It initiated an interdisciplinary exchange about existing work, open problems, and interesting applications.

Seminar March 15–20, 2015 – http://www.dagstuhl.de/15122
1998 ACM Subject Classification F.4.2 Grammars and Other Rewriting Systems, F.4.3 Formal Languages, G.2.2 Graph Theory, I.2.7 Natural Language Processing
Keywords and phrases Automata theory, Graph transformation, Natural language processing
Digital Object Identifier 10.4230/DagRep.5.3.143
Edited in cooperation with Petter Ericson (pettter@cs.umu.se)

1 Executive Summary

Frank Drewes
Kevin Knight
Marco Kuhlmann

Strings are fundamental data structures in natural language processing (NLP). Weighted finite-state string acceptors and transducers, first introduced as theoretical constructs, have proven their worth in speech recognition, part-of-speech tagging, transliteration, and many other applications. The string automaton framework provides efficient generic algorithms for composition, bidirectional application, k-best extraction, determinization, minimization, parameter tuning, etc. These algorithms have been packaged in software toolkits that form the core of many state-of-the-art systems.

Tree automata go further in permitting large-scale, syntactically-motivated re-ordering of subtrees. They were originally devised to help formalize Chomsky's linguistic theories, but

Formal Models of Graph Transformation in Natural Language Processing, *Dagstuhl Reports*, Vol. 5, Issue 3, pp. 143–161
Editors: Frank Drewes, Kevin Knight, and Marco Kuhlmann

Dagstuhl Reports
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

their subsequent development was largely disconnected from NLP practice. In 2005, tree automata theorists and machine translation (MT) practitioners began working together to come up with a new kind of statistical MT system based on tree automata. This led to some of the best practical results in common evaluations of MT quality, and syntactic methods are now used in industrial MT systems. This work at the intersection of tree automata and NLP created vibrant new research directions for both areas.

Nowadays, *graphs* are becoming an even more general fundamental data structure in practical NLP. Classic feature structures can be seen as rooted, directed, edge- and leaf-labeled graphs. Recent work in dependency parsing produces graphs rather than trees. New work in deep semantic annotation organizes logical meanings into directed graph structures, and several efforts are now being made that in the near future will yield large amounts of linguistic data annotated with these representations. Formal models of *graph transformation* are therefore of fundamental importance for the development of practical systems for these tasks. The situation is familiar: there exists a formal theory of graph transformation, but this theory is largely disconnected from research and practice in NLP.

The theory of graph transformation studies rule-based mechanisms for the manipulation of graphs. A particularly well-studied subject within the area of graph transformation, and one that has received quite some attention recently within the NLP community, are *context-free graph grammars*. These grammars have many nice properties in common with context-free phrase structure grammars, but are considerably more powerful and versatile; in particular, they can be used to generate context-sensitive string languages (when strings are represented as chain graphs). The price of this expressiveness is a higher computational complexity; in particular, there are context-free graph languages for which parsing is NP-complete. This has triggered research on specialized, more efficient algorithms for restricted classes of graphs. A well-known result in this area is that many in general intractable problems on graphs become solvable in polynomial time when restricted to graphs of bounded tree-width.

With the number of interesting applications and the amount of available data quickly increasing, there is a clear need for the NLP community to acquire knowledge about formal models of graph processing, as such models can greatly simplify practical systems, by providing a uniform knowledge representation and efficient, generic algorithms for inference. Unfortunately, most NLP researchers are unaware of the rich literature on graph transformation, and even those who are find it hard to connect it to their own work. Conversely, few researchers in graph transformation are aware of the new applications of their research within natural language processing, the characteristic properties of the available data, the specific desiderata of these applications, and the research problems that are posed by them.

The overall goal of the seminar was to bring the various research communities together to assess the state of the art, identify areas of common interest, and pave the way for future collaborations. We think that this goal was reached to a very high degree, which will be a major factor in the creation of a new interdisciplinary research community.

Organization of the Seminar

The seminar was attended by 29 participants from 9 countries in North America, Europe, and Africa. It was held from March 15 to March 20, 2015. Since the intention was to foster the creation of a new research community, it was decided to organize the seminar in the form of a self-organized workshop with many informal discussion meetings on topics suggested by the participants themselves. For this, the seminar roughly followed the idea of Open Space Technology. This worked very well and gave rise to many insightful discussions. (See Section 5 for the list of topics discussed.)

2 Table of Contents

3 Overview of Talks

3.1 Tutorial: Introduction to Graph Transformation

Frank Drewes (University of Umeå, SE)

URL http://www8.cs.umu.se/~drewes/gratra_tutorial.pdf

The theory of graph transformation studies formal rule-based models for the manipulation of graphs. In particular, this includes graph grammars that generate graph languages and graph rewrite systems that turn input graphs into output graphs. The tutorial gave an introduction to some of the most well-studied aspects of graph transformation, focusing on those which seem to be of particular interest for NLP.

General Graph Transformation Systems

These are graph transformation systems which are Turing complete. They usually consist of finitely many rules (possibly enhanced by control structures or application conditions) that replace a subgraph (the left-hand side) in a host graph by another graph (the right-hand side). The most well-known approaches are the single and double pushout approaches, which belong to the so-called algebraic approaches.

Context-Free Graph Grammars

Context-free graph grammars are grammars based on rules that either replace single nodes or single (hyper)edges by other subgraphs. This makes them context-free and results in many desirable properties, e.g., there exist algorithms for various tasks.

Parsing Hyperedge Replacement Languages

One of the most important algorithmic tasks in connection with context-free graph grammars, such as hyperedge replacement grammars, is parsing. There are easy and very versatile proofs showing that this problem is NP-hard even in the non-uniform case, i.e., where the grammar is fixed. In other words, there are NP-complete hyperedge replacement languages. However, in special cases polynomial-time parsing is known to be possible.

Monadic Second-Order Logic

There are well-known and very useful connections between monadic second-order logic on strings or trees on the one hand, and regular string and tree languages on the other hand. Similarly useful connections relate monadic second-order logic on graphs with context-free graph languages. For example, the restriction of a context-free graph language by a logical sentence is again context-free, and it can be decided whether all graphs/finitely many graphs/no graphs of a given context-free graph language satisfy a given sentence.

Term Graphs

Acyclic directed graphs can represent terms with sharing, so-called term graphs. This has been used in order to implement functional programming languages efficiently. Since directed acyclic graphs are important in meaning representation, the techniques and results of term graph rewriting may turn out to be useful in NLP.

4 Reports from Working Groups

4.1 Convolution Kernels for Graphs

Giorgio Satta

In machine learning, *kernels* are a class of functions that measure the "similarity" between two objects or structures. Many algorithms implementing kernel functions use an underlying feature vector representation for the input structures in a space with very high or even infinite dimension, but only implicitly represent this feature space. The advantage is that kernel algorithms avoid explicit computation of the feature map in such large space, and are thus much more efficient than direct algorithms.

Convolution kernels use a decomposition of the input structures into overlapping substructures or patterns, and are based on the computation of a census function for these substructures. In natural language processing, convolution kernel functions have been introduced for strings and trees. This group has been exploring existing convolution kernel methods for graph structures, in view of their potential application to semantic analysis of natural language based on directed acyclic graph structures.

Several convolution kernels for graphs have been proposed in the literature, based on simple path or tree-like substructures; see [1] and references therein.

This group has been exploring the idea of using so-called elastic kernels for directed acyclic graphs. The idea is borrowed from tree-based kernels as used in syntactic parsing (Alessandro Moschitti, personal communication). In an elastic kernel, substructure matching is allowed by stretching an arc of the pattern graph over a path of several arcs in the graph under analysis. This allows "jumping" over portions of the semantic representation that might correspond to some adjunct or modifier. Linguistic relevance of this idea has been discussed, without reaching a consensus among the participants on its effectiveness in semantic representations.

This group has also looked into existing similarity measures for Abstract Meaning Representation (AMR) such as smatch [2]. It has been observed that the smatch similarity measure does not satisfy the standard kernel function conditions and therefore can not be considered a well-formed kernel. This is so because convolution kernels make use of summations over substructure matching, while smatch make use of a max operator over the same counts.

References

1 Nino Shervashidze and Karsten M. Borgwardt. Fast subtree kernels on graphs. In Yoshua Bengio, Dale Schuurmans, John D. Lafferty, Christopher K. I. Williams, and Aron Culotta, editors, *Advances in Neural Information Processing Systems 22: 23rd Annual Conf. on Neural Information Processing Systems 2009. Proc. of a meeting held 7–10 December 2009, Vancouver, British Columbia, Canada*, pp. 1660–1668. Curran Associates, Inc., 2009.

2 Shu Cai and Kevin Knight. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

4.2 Efficient HRG-Parsing for the NLP Domain

Christoph Teichmann and Frank Drewes

Hyperedge Replacement Grammars (HRGs) are one of the candidate formalisms for the generation and parsing of graph based semantic representations [1]. In general, HRGs can generate NP-complete languages [2, 3]. Thus, one cannot hope to develop efficient parsing algorithms that work for all HRGs.

While it has been known for a long time that parsing based on HRGs can theoretically be implemented efficiently if the graphs that are being processed exhibit certain properties [4], researchers are currently trying to solve many of the details of actual implementation. Parsing algorithms often require a number of very complex design decisions in order to be efficient.

One approach to more efficient parsing has been the proposal to make use of tree decompositions of rules and input graphs, and work on representations based on the "boundary" of subgraphs that have already been processed [5]. This approach is asymptotically more efficient than storing copies of complete subgraphs and can be used to establish upper bounds on the number of items that need to be considered. On the comparatively small graphs that are used as input data in natural language processing, however, it may actually be more efficient just to store the complete subgraphs.

Once the questions of representation and look-up have been settled, it is then possible to design graph parsers in a way that is very similar to well known approaches in string parsing for natural language processing.

Another approach that is currently being worked on is to generalize techniques known from compiler construction, based on restrictions that make string parsing more efficient than the usual $O(n^3)$ obtained by CKY or Earley parsing. A first approach in this direction is *predictive top-down parsing*, which extends SLL(1) string grammars to the HRG case. A predictively top-down (PTD) parsable HRG yields a quadratic, and in many cases linear parsing algorithm. However, the analysis of a grammar needed to establish PTD parsability is complicated and cannot usually be done by hand [6]. It is currently still unclear whether PTD parsable HRGs are suitable for NLP applications. It may be worthwhile to check whether predictive bottom-up parsing is possible as well, generalizing the well-known notion of LR(1) string parsing.

Another idea, which may be especially useful for parsing structures such as Abstract Meaning Representations, is to extend Lautemann's almost forgotten concept of componentwise derivations [4]. Roughly speaking, the intuition behind componentwise derivations is that, if a nonterminal generates a graph that consists of several connected components, then the derivations of the individual components are independent of each other. This corresponds well to the intuition that, if several modifiers are attached to a concept in an AMR, then the sub-DAGs corresponding to those modifiers can usually be generated independently.

References

1 Bevan Keeley Jones, Sharon Goldwater, and Mark Johnson. Modeling graph languages with grammars extracted via tree decompositions. In *Proceedings of the 11th International Conference on Finite State Methods and Natural Language Processing*, page 54–62, 2013.

2 I. J. Aalbersberg, A. Ehrenfeucht, and G. Rozenberg. On the membership problem for regular DNLC grammars. *Discrete Applied Mathematics*, 13:79–85, 1986.

3 Klaus-Jörn Lange and Emo Welzl. String grammars with disconnecting or a basic root of the difficulty in graph grammar parsing. *Discrete Applied Mathematics*, 16:17–30, 1987.

4 Clemens Lautemann. The complexity of graph languages generated by hyperedge replacemen. *Acta Informatica*, 27:399–421, 1990.

5 David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. Parsing graphs with hyperedge replacement grammars. In *Proc. of the 51st Annual Meeting of the Association for Computational Linguistics*, page 924–932, 2013.

6 Frank Drewes, Berthold Hoffmann, and Mark Minas. Predictive top-down parsing for hyperedge replacement grammars. In F. Parisi-Presicce and B. Westfechtel, editors, *Proc. 8th Int'l Conf. on Graph Transformation (ICGT'15)*, Lecture Notes in Computer Science. Springer, 2015.

4.3 Grammarless Approaches

Joakim Nivre, Christoph Teichmann, and Giorgio Satta

The term *grammarless approaches* is used informally to describe parsing algorithms where no hard constraint is imposed on the search for a syntactic analysis of the input sentence. In other words, the parser never rejects any input sentence. Typically, in a grammarless approach the parser uses soft constraints (weights) to choose some syntactic analysis in a space that includes all candidate structures that are compatible with the input sentence. More specifically, some search process is carried out that applies soft constraints either globally or locally to optimize possible choices.

In dependency parsing a number of transition systems and weighting schemes have been proposed that can generate projective and/or non-projective dependency trees given an input sentence. It is possible to use these to efficiently generate analyses for input sentences, when they are paired with a classifier that selects one out of a list of possible transitions given the current state of the parsing process and the input data, or by finding maximum weight substructures. It seems natural to extend this approach to structures that do not obey the treeness constraint.

In this group we discussed transition systems and weighting schemes capable of selecting different graph structures and their comparative benefits. In this context it is important to consider the problem of learning the weight function that guides the selection of the final analysis and whether it is possible to efficiently reconstruct the operations that generated a graph. It was observed that there is a simple generalization of Covington's algorithm for dependency tree parsing [1] that can derive any graph in time quadratic in the length of the input sentence. This algorithm could possibly be restricted to interesting special cases in order to improve accuracy or efficiency or both.

Generation/Decomposition of Graphs with Page Number Bounded by Two

In graph theory, a book embedding of a graph is an embedding into a collection of half-planes, all having the same line as their boundary. The vertices of the graph are required to lie on this boundary line, and the edges are required to stay within a single half-plane. The *page number* of a graph is the smallest possible number of half-planes for any book embedding of the graph [2].

This group has focused on dependency graphs with page number of two, where the vertices of the graph lying on the book boundary line is fixed and specified by the input sentence. The group has then discussed the problem of computing the highest score dependency graph with page number of two given an input sentence. Note that here we are considering dependency structures that are graphs in the strict sense, that is, these structures are not tree-like graphs.

Discussion has focused on how to apply dynamic programming techniques to solve this problem efficiently, therefore showing that the problem is in PTIME. There are known dynamic programming algorithms for the case of dependency graphs with page number of one; see [3]. Shortly after the meeting in Dagstuhl, two participants to this discussion group came up with a proof that the problem at hand is NP-hard; see [4].

Grammarless Generation of Strings from DAGs

We also discussed the problem of generating strings from DAGs, which can be viewed as a kind of graph transduction problem. The question then becomes: is there a grammarless approach to transducing graphs? Informally, the answer seems to be yes, and it is helpful to first think about how grammarless parsing algorithms apply to strings and trees. In the string case, the set of outputs can be defined as the set of all trees over a set of input words. In some cases we define it more carefully as the set of projective trees, or the set of non-projective trees meeting particular criteria [5]. We can then use combinatorial optimization algorithms or transition systems to search over the set of output trees.

Before considering generation of strings from graphs, we can consider generation of strings from trees, which is a special case (this is sometimes called realization in the generation community). Suppose that we have a labeled input tree. Then the output might be defined as follows: the string obtained by any tree traversal and substitution operation on the nodes of the tree. In other words, at each node of the tree, we must visit the node and its children (recursively) once, and when we visit the node, we output a word. This can be thought of as linearization of an (unordered) dependency tree, which always results in a projective structure. It should be possible to produce non-projective structures by encoding them into the traversal [6]. To extend this idea to DAGs, it suffices to observe that some nodes can be visited more than once, but we always know the number of visits since we know the number of parents. Hence, we can split the (recursive) visit of a node into as many parts as there are parents, and execute each part in order as the node is visited from its parents. Algorithmically, this could be accomplished many ways: by a transition system that outputs words as it visits nodes, or a global or local model that predicts the visit order for each node.

References

1 Michael A Covington. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th annual ACM southeast conference*, pages 95–102. Citeseer, 2001.

2 Frank Bernhart and Paul C Kainen. The book thickness of a graph. *Journal of Combinatorial Theory, Series B*, 27(3):320–331, 1979.

3 Marco Kuhlmann. Tabulation of noncrossing acyclic digraphs. CoRR abs/1504.04993, Linköping University, 2015.

4 Peter Jonsson and Marco Kuhlmann. Maximum pagenumber-k subgraph is NP-complete. CoRR abs/1504.05908, Linköping University, 2015.

5 Emily Pitler, Sampath Kannan, and Mitchell Marcus. Finding optimal 1-endpoint-crossing trees. *Transactions of the Association for Computational Linguistics*, 1:13–24, 2013.

6 Marco Kuhlmann and Matthias Möhl. Mildly context-sensitive dependency languages. In *Proc. Association for Computational Linguistics*, pages 160–167, 2007.

4.4 HRG-Grammar Induction for NLP

Christoph Teichmann

Recently there has been interest in generating graph based semantic representations for input sentences and/or generating sentences that correspond to the meaning represented by a graph. This task can be solved by using synchronous grammars that generate a string and a graph in parallel. One can then parse an input with one side of the synchronous grammar and use any result that would have been generated by to other side as output. This leads naturally to the problem of inferring these synchronous grammars from input data. Since we will usually only have access to sentences and graphs and not to any information about an underlying grammar, we are forced to consider a potentially large set of parallel derivation steps and then extract a – preferably small – grammar. We assume that the sentence has been generated by some context-free derivation tree and that the graph was generated by a hyperedge replacement derivation. Unfortunately there are a large number of many potential pairings of sentence and graph parses, even when they are only considered in some packed representation. Therefore it is necessary to employ alignments that are the result of some simpler pre-processing step.

4.5 Probabilities for DAG Automata and Other Non-Context-Free Graph Rewriting Systems

Adam Lopez

The last few decades of work in natural language processing have confirmed that probabilistic systems learned from data are indispensable. So graph rewriting systems used for natural language processing must be probabilistic. Consider semantics-based machine translation, in which the goal is to explicitly convert a source string to its semantic representation, and then convert this representation to a target string, as in the following example from Jones et al. [1].

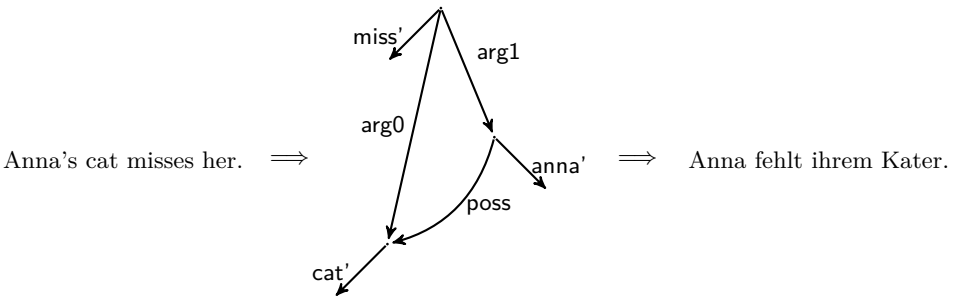

Figure 1 Example translation using semantics.

In a probabilistic setting, our goal is first to predict a graph g from a source string s, and then to predict a target string t from g, giving us a model $p(t, g|s) = p(t|g)p(g|s)$. Jones et al. [1] suggest solving this with a pair of synchronous grammars, each defining a probabilistic relation

on string/ graph pairs. Given a language of source strings $\mathcal{L}_s$, a language of source graphs $\mathcal{L}_g$, a language of target graphs $\mathcal{L}_{g'}$ and a language of target strings $\mathcal{L}_t$, these grammars define relations $R \subseteq \mathcal{L}_s \times \mathcal{L}_g$ and $R' \subseteq \mathcal{L}_{g'} \times \mathcal{L}_t$. From these relations, we can formally define translations as the sets of semantically equivalent strings: the set of all translation pairs is $\{s, t | \exists g : s, g \in R \wedge g, t \in R'\}$. Hence we must be able to efficiently compute the intersection of the graph languages, $\mathcal{L}_g \cup \mathcal{L}_{g'}$. Compositions of this kind are widely used on string data in current speech and machine translation models, where they can be implemented using compositions of finite-state transducers [2, 3].

Our goal is to define similarly composable probabilistic graph languages. If our graph grammar is context-free (in the sense that its productions are associative and commutative), then we can attach normalized weights to each production to define a probability distribution over the set of graphs that it generates. We can also define its productions to be isomorphic to a (string) context-free grammar, enabling us to define probabilistic relations on strings and graphs, as desired. Although grammars and automata on graphs are much less well-studied than they are on strings and trees, two candidate formalisms have recently been identified: *hyperedge replacement grammar* (HRG), described by Drewes et al. [4] and studied by Chiang et al. [5]; and *DAG automata*, introduced by Kamimura and Slutzki [6] to model type-0 derivations and studied by Quernheim and Knight [7].

Unfortunately, neither HRG nor DAG automata satisfy our desired criteria. Although HRGs are context-free and can easily be made probabilistic, they are not closed under intersection – the emptiness of intersection is undecidable, as are many other useful questions on HRGs. In contrast, DAG automata are closed under intersection but are not context-free –so it is unknown how to make them probabilistic in a way that would yield practical algorithms, although weighted algorithms have been developed that do not define proper probability distributions. The session addressed the question of whether a fully probabilistic treatment of DAG automata is possible. Informally, the fundamental problem stems from a confluence of properties:

1. DAG automata were initially designed to model type-0 derivations, so they are non context-free. This means that rewriting operations are not commutative: applying a single rewrite to the frontier states of a DAG automaton may change the set of rewrites that are applicable to remaining states.
2. One plausible way to define probabilistic rewriting systems is to define a probability distribution over all possible rewriting steps that can be applied to a particular configuration of the system. However, property 1 implies that this probability distribution cannot factor over the frontier states of the automaton: the rewriting of any particular state is not independent of rewriting other states. Most likely, this means that a probability distribution over derivations of a DAG automaton cannot factor over its productions, as it can in context-free formalisms. So, probability distributions must depend on the complete configuration of the automaton.
3. Since probability depends on the state of the automaton, steps of a probabilistic parsing algorithm must also know the state of the automaton before and after application of a particular rewrite. Unfortunately, the naïve algorithm for this is to define a parsing state for every cut of an input graph, and the number of cuts is exponential.

Hence, it appears that a natural definition of probabilistic graph automata leads to exponential probabilistic recognition algorithms, which is actually worse than in the weighted case. It is an open problem whether a better solution is possible.

References

1 Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Mortiz Hermann, and Kevin Knight. Semantics-based machine translation with hyperedge replacement grammars. In *Proceedings of COLING*, 2012.

2 Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. Speech recognition with weighted finite-state transducers. In Larry Rabiner and Fred Juang, editors, *Handbook on Speech Processing and Speech Communication, Part E: Speech recognition*. Springer, 2008.

3 Cyril Allauzen, William Byrne, Adria de Gispert, Gonzalo Iglesias, and Michael Riley. Pushdown automata in statistical machine translation. *Computational Linguistics*, 2014.

4 Frank Drewes, Hans-Jörg Kreowski, and Annegret Habel. Hyperedge replacement graph grammars. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, pages 95–162. World Scientific, 1997.

5 David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. Parsing graphs with hyperedge replacement grammars. *Proceedings of ACL*, 2013.

6 Tsutomu Kamimura and Giora Slutzki. Parallel and two-way automata on directed ordered acyclic graphs. *Information and Control*, 49(1):10–51, 1981.

7 Daniel Quernheim and Kevin Knight. Towards probabilistic acceptors and transducers for feature structures. In *Proceedings of the Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2012.

4.6 Properties of (Various Types of) DAG Automata

Frank Drewes

Graph structures in NLP are often directed acyclic graphs (DAGs). In particular, representations of meaning such as Abstract Meaning Representations (AMRs) may safely be assumed to be DAGs. Therefore, automata working on DAGs, similar to the well-known concept of tree automata, could be of great usefulness if they (a) allow to recognize DAG languages that are of interest from an NLP point of view and (b) exhibit useful closure properties as well as algorithmic properties. Ideally, such a theory should also provide corresponding notions of transducers, i.e., automata with output. Unfortunately, not much work seems to have been done in this field. The oldest approach [1, 2] was explicitly invented to capture the nature of derivations in type-0 Chomsky grammars and is thus much to powerful. The approach of [3, 4] considers only DAGs that are trees with maximally shared subtrees. This is clearly inappropriate for processing AMRs or other representations of meaning because such DAGs cannot contain isomorphic sub-DAGs. An interesting approach from the NLP point of view seems to be the one proposed in [5], but it has some disadvantages:

- It is capable of generating NP-complete DAG languages, and thus parsing is too inefficient.
- The recognized DAG languages have non-context-free path languages in the worst case, whereas it is reasonable to assume that the set of all valid meaning representations (in any reasonable formalism such as AMRs) have regular path languages.
- The class is not closed under complementation.

It is unclear how important the last point is, but from a formal point of view closedness under complementation would certainly be a positive property. Hence, it seems that [5] may

be used as a starting point and source of inspiration, but does not provide a satisfactory solution in itself.

Properties such as those mentioned above left aside, there are properties of the DAGs being worked on that are of interest:

- Incoming and outgoing edges of a node in a DAGs may be ordered or unordered. In the ordered case, this order may be defined globally by placing an order on the nodes of a DAG. This type of DAGs is considered in [1, 2], and it seems that the global order (and the fact that it defines the local order at each node) is responsible for the power of these automata. From the point of view of NLP, and in particular from the point of view of AMRs, it seems that outgoing edges of a node should at least be partially ordered (or labeled, which is equivalent) whereas incoming edges should be unordered. Thus, an ideal model should be able to capture both.
- DAGs can be ranked or unranked, where ranked means that every node has a predefined number of incoming and outgoing edges determined by its label. Meaning-representing DAGs are usually unranked in the sense that there is no a priori bound on the number of incoming and outgoing edges of a node. From this point of view, unranked seem to be more appropriate. However, given the fact that it is difficult to devise a model that is both general and has nice properties, it may be reasonable to consider the ranked case, anyway, at least as a first step towards a more general model.

More recently, Quernheim and Knight [6] proposed a new notion of DAG automata based loosely on the approach by Kamimura and Slutzki. Inspired by that approach, Chiang, Drewes, Gildea, Lopez, and Satta have started to work on a restricted model of ranked DAG automata that has been discussed during the Dagstuhl Seminar. These automata have a decidable emptiness problem and regular path languages, the latter being an insight obtained during the discussions of the Dagstuhl Seminar (with considerable help of J. Björklund and A. Maletti). Unfortunately, even these rather restricted DAG automata can recognize NP-complete DAG languages.

References

1 T. Kamimura and G. Slutzki. Parallel and two-way automata on directed ordered acyclic graphs. *Information and Control*, 49:10–51, 1981.

2 T. Kamimura and G. Slutzki. Transductions of dags and trees. *Mathematical Systems Theory*, 15:225–249, 1982.

3 W. Charatonik. Automata on DAG representations of finite trees. Research Report MPI-I-1999-2-001, MPI Saarbrücken, 1999.

4 S. Anantharaman, Narendran P., and M. Rusinowitch. Closure properties and decision problems of dag automata. *Information Processing Letters*, 94:231–240, 2005.

5 L. Priese. Finite automata on unranked and unordered DAGs. In *Proc. 11th Int'l Conf. on Developments in Language Theory (DLT 2007)*, volume 4588 of *Lecture Notes in Computer Science*, pages 346–360, 2007.

6 Daniel Quernheim and Kevin Knight. Towards probabilistic acceptors and transducers for feature structures. In *Proc. 6th Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 76–85. Association for Computational Linguistics, 2012.

4.7 Typical or Desirable Features of Graphs in NLP

Stephan Oepen, Mark Steedman, Frank Drewes, Laura Kallmeyer, and Daniel Bauer

GraphaLogue: Surveying Graph Banks

With a growing community interested in graph processing, it would seem worthwhile to compile a survey of existing graph banks, i.e. collections that pair natural language data with graph-structured representations of linguistic analysis (syntactic, semantic, or otherwise). Initially at least, such resources will primarily be annotations of meaning (in various interpretations), but in principle other types of linguistic annotations that transcend tree-structured analyses should be included. For this survey, one should define and quantify relevant structural properties such as:

- reentrancy
- edge density
- connectedness
- rootedness
- acyclicity
- functionality of edge labels
- treewidth
- page number

Likewise, it would be helpful to try and characterize the (purpose and) contents of the annotations, sentence and token counts, and licensing.

Besides providing a catalogue of available resources, this survey could also develop into a quantitative and qualitative comparison of representations. To the extent that we can tease apart different layers of semantic construction – for example differentiate grammatical control from anaphoric binding – it would also seem useful to characterize annotated resources in terms of which of these phenomena they target.

Marco Kuhlmann and Stephan Oepen would be happy to try and coordinate an initial catalogue (or "graphalogue", if you will) construction. With a bit of luck, this could evolve into a community resource (e.g. on the ACL Wiki) and enjoy collective maintenance over time. Obvious existing resources to look at include:

- AMR Bank
- SemEval 2014 and 2015 Semantic Dependency Parsing (SDP) graphs
- Universal Dependencies
- Semantic Dependencies in CCGBank

Semantic Representations Adapted to Inference

A central problem in using semantic parsing for tasks like open-domain question answering and (more obviously) machine translation is that the sentence in unseen text or the target language may take a form that is not the one most directly suggested by the question or source. (Thus the answer to the question "Did Google buy YouTube" may or may not be answered by "Google bought every company", "Google's purchase of YouTube", "Google subsidiary YouTube," "L'acquisition de YouTube par Google", etc.) Most semantic parsers are too specific to the original form of language to allow the question to be settled without lengthy inference of a kind that is not usually affordable – hence the habit of search engines of returning multiple pages containing such phrases in the hope that the user can work out

the answer by reading them. Often the user can do this, but sometimes they cannot. (Try "What are Miles Davis Recordings without Fender-Rhodes piano?").

Since the Generative Semantics and Conceptual Dependency Semantics of the '70s, there have been many attempts to produce a Universal or "Natural" Semantics underlying paraphrase and common-sense entailment relations between expressions, including attempts to use such a representation or "Interlingua" for Machine Translation. However, none of them have got very far beyond language specificity, even when multiple languages have been considered, and there has recently been a move to recast the problem in machine language terms as that of learning a "hidden" set of semantic relations from large amounts of unannotated text.

Two main approaches were distinguished. The most radical is the "pure distributional" approach, based on collocations of content words represented as dimensionally reduced vectors, with linear algebraic operations like vector addition and multiplication substituting for traditional semantic composition in forming meanings for larger structures, often under the control of dependency parsers [1]. Such representations have some striking advantages, such as being able to simultaneously represent multiple ambiguous readings, which may be disambiguated by linear algebraic composition.

Such representations are capable of representing the similarity of concepts as closeness in the multidimensional vector space, and hence of detecting the similarity between paraphrases in source and target. However, it is hard to see how they can be interfaced with logical semantics. In particular, there does not seem to be a vector or linear algebraic representation for operators such as negation. A second kind of distributional semantic seeks to identify relations of paraphrase and entailment directly in unseen text, using parsing or "machine reading", and to build such logical relations into natural language semantics directly, treating paraphrases as clusters and entailment as logical conjunction [2].

The latter approach has been shown to to be capable of capturing linguistically significant entailments, such as that "McCain regrets that he wasn't nominated" entails that "McCain wanted to be nominated", which could be used to acquire the information that the semantics of verbs like "want" includes an implicit controlled subject of the complement "to be nominated".

There was further discussion of the vector based alternative, and whether recent developments using "Deep Learning", or multi-layer perceptrons or Boltzmann machines using backpropagation training at a vast scale would render interaction with logicist semantic unnecessary. It was generally felt that the radical approach was probably incompatible with any form of structured representation such as AMR, but that the paraphrase and entailment based clustering approaches were entirely compatible and might even be helpful.

Logical Operators and Quantification

Currently logical operators and quantification are not represented explicitly in most graph based meaning representations like AMR, although they have been used traditionally to represent linguistic meaning. We discussed if we can and should develop a graph-based representation that has a translation into some logical form. Such a representation would allow for semantic inference but can be difficult to annotate, especially when annotators have to resolve possible scope orders.

In AMR, some vertices represent existentially quantified variables (instances of concepts, such as events and objects), which also interferes with the scope order. Universal quantification and negation is expressed using additional edges. Negation attaches to the root of the sub-DAG it takes scope over. Disjunction is represented using an additional "or" node. Conjunction is sometimes represented this way, but only if it is mentioned explicitly in the

sentence (for instance, conjunction of two events). Unfortunately scope is not adequately represented in the structure of the AMR graph. Logical operators are generally only represented if they are mentioned explicitly in the sentence described by the AMR.

Existentially quantifying all event nodes allows us to account for the reading with narrow scope of the existentially quantified event in the following sentence: *"Most of the students have read the book."* In this reading there is one *reading* event for each student. Without an explicit representation of scope the reading in which the existential quantifier takes broad scope is lost. According to this reading there is a single *reading* event of the book, for instance if the students take turns.

A better solution might be to represent scope outside the graph structure, on a separate representation level. Scope could then be left underspecified if it is ambiguous. Annotators could either annotate the specific order of quantifiers, for instance

"Most people know two languages": most people > exists knowing > two languages

or they could specify dependencies in Skolem terms, such as

"most people know two languages": language(people), know(most)

The second option appears to be more intuitive for the annotator and it would allows annotators to leave out dependencies they are not sure of (or that are actually ambiguous or independent of each other). The result would be an underspecified representation of quantifier scope that allows for reasoning.

We leave the specification of a graph-based representation that addresses these issues and an annotation scheme for future work.

Linguistic Phenomena that "Cause" Reentrancies in AMRs

The following "causes" of reentrancies could be discovered by looking at a variety of AMRs:

- anaphora such as pronouns
- control (-like) structures such as in *"John promised me to paint the wall."* (John will be the $\texttt{arg}_0$ of `paint`.)
- multiple participles (`NB` "front-loading" assigns $\texttt{arg}_1$)
- VP coordination / shared conjuncts
- implicit arguments ("he" is $\texttt{arg}_1$ of "hospital treatment")
- relative clauses

Argument Sharing Exemplars

Besides inspecting concrete example annotations in the AMR bank, a more general inventory of phenomena that cause reentrancies in semantic graphs due to argument sharing is of interest. We aim at creating such a collection of argument sharing exemplars that will be a useful resource for linguistic analysis and grammar developing. The following list is a first collection of such phenomena.

- *Grammatical Control*
  Kim wants to sing. ; subject-equi
  Kim wants Sandy to sing. ; raising-to-object (no reentrancy)
  Kim persuaded Sandy to sing. ; object-equi
  Sandy seemed to sing. ; raising-to-subject (no reentrancy)
  Kim promised to seem to be competent.

- *Passives*
  Kim wants to be heard.
- *Nominalizations*
  Kim made a promise to sing.
  Kim has a desire to sing.
  Kim's plan is to sleep more.
  Kim showed signs of recovery.
- *Modification*
  The drying and washing machine broke.
  The washing machine is expensive.
- *Coordination*
  Kim drank wine and ate pizza.
  Kim showed Sandy and sold Tony the wine.
  Kim showed and Sandy sold the wine.
  Kim sold the wine and Tony the pizza.
  Kim and Sandy sang.
  his arms and feet. ; interaction with possessive determiner
  Kim sang on Monday and on Tuesday.
  Kim wanted and expected to sing.
- *Reflexive Pronouns and reciprocals*
  Kim saw herself.
  Kim and Sandy admired each other.
- *Relative Clauses*
  Kim ate the pizza that Tony had sold.
  Kim saw the boy whose father sold the pizza.
  Kim arrived on the day that Sandy arrived.
- *Secondary Predicates*
  Kim placed the book on the table.
  Kim wiped the table clean.
  Kim left Sandy without paying.
  Kim met Sandy singing.
  Kim met Sandy drunk.

Stephan Oepen and Laura Kallmeyer plan to extend this initial list of examples in the near future to a more complete resource called *SemSharE – Sem*antic Argument *Shar*ing *E*xemplars.

References

1 Sebastian Padó and Mirella Lapata. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199, 2007.
2 Mike Lewis and Mark Steedman. Combining distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192, 2013.

5 Seminar Program

Introductory Presentations

The seminar started by two introductory presentations on the use of graphs for representing meaning:

1. Kevin Knight. *Mapping English Strings to Reentrant Semantic Structures*
2. Marco Kuhlmann. *Properties of the SemEval-2015 Data Sets*

Tutorial on Graph Transformation

In addition, a longer tutorial on the theory of graph transformation, divided into several parts and spread out over 3 days, was given by Frank Drewes.

Working Groups

Open Space group discussions were held on Monday, Tuesday, and Thursday. The following topics were discussed:

Monday

Session 1

- Node replacement grammar for semantics. What does treewidth mean (in NLP)?
- (Greedy) parsing algorithms for graphs. Restricted Graph formalisms that allow efficient parsing. Grammarless parsing
- DAG automata. Generating and recognizing AMRs
- Universal dependencies. Syntax–semantics interface. Integrating logical operators into semantic graphs. Comparison of graph banks

Session 2

- Characterizing graphs produced by various grammar formalisms. Tree-to-DAG transformations
- What happens to HRG when we impose a linear order on the nodes?
- Identifying a "good" generator set of graph operations
- Multitape graph transducers (insufficient output)
- Convolution kernels for directed acyclic graphs and other similarity measures

Tuesday

Session 1

- DAG automata. Generating and recognizing AMRs
- Grammarless approaches to graph parsing
- Dress up an inventory of the graphs we want to have; what are the consequences for required HRG?
- Hyperedge Unification Grammars

Session 2

- Characterizing graphs produced by various grammar formalisms. Tree-to-DAG transformations
- Practical parsing of HRG
- Grammarless generation

Thursday

Session 1

- Types and causes of reentrance. Linguistically relevant graph grammars. An inventory of the graphs that we want to have; consequences regarding constraints on HRG
- An argmax-algorithm for pagenumber-2 graphs. Relationship to dependency parsing. Eisner-like algorithms and treewidth (or other notions of x-width)

Session 2

- How can semantic representations support inference?
- Restricted but fast HRG parsing

Session 3

- Graphalog – a catalogue of graph banks/Example-based comparisons of graph representations (AMR, SDP etc.)
- Inducing synchronous context-free string $\leftrightarrow$ graph transformations. Graph-string alignment algorithms

Evening session

- How do we assign probabilities to graphs? Probabilistic non-context-free graph rewriting

Note that the remainder of this report is not structured according to the list above. Instead, we have tried to structure the major outcomes of the discussions and present them in an appropriate way in order to serve as a reference for future work.

Closing Session

Friday morning was devoted to a general recap and an evaluation of the seminar. The result of the evaluation was very positive; it was decided to consider the possibility of applying for a follow-up workshop after a couple of years when the community and its research area had taken shape, which to a significant extent would be thanks to this Dagstuhl Seminar.

Participants

- Daniel Bauer
Columbia University, US
- Suna Bensch
University of Umeå, SE
- Henrik Björklund
University of Umeå, SE
- Johanna Björklund
University of Umeå, SE
- David Chiang
Univ. of Notre Dame, US
- Frank Drewes
University of Umeå, SE
- Petter Ericson
University of Umeå, SE
- Daniel Gildea
University of Rochester, US
- Karl Moritz Hermann
Google DeepMind – London, GB
- Berthold Hoffmann
Universität Bremen, DE
- Peter Jonsson
Linköping University, SE
- Laura Kallmeyer
Heinrich-Heine-Universität Düsseldorf, DE
- Kevin Knight
USC – Marina del Rey, US
- Alexander Koller
Universität Potsdam, DE
- Marco Kuhlmann
Linköping University, SE
- Adam Lopez
University of Edinburgh, GB
- Andreas Maletti
Universität Stuttgart, DE
- Jonathan May
USC – Marina del Rey, US
- Mark Minas
Universität der Bundeswehr – München, DE
- Joakim Nivre
Uppsala University, SE
- Stephan Oepen
University of Oslo, NO
- Detlef Plump
University of York, GB
- Giorgio Satta
University of Padova, IT
- Natalie Schluter
University of Copenhagen, DK
- Mark Steedman
University of Edinburgh, GB
- Christoph Teichmann
Universität Potsdam, DE
- Brink van der Merwe
University of Stellenbosch, ZA
- Heiko Vogler
TU Dresden, DE
- Daniel Zeman
Charles University – Prague, CZ