Chapter 1

k-Valued Link Grammars are Learnable from Strings

DENIS BÉCHET

ABSTRACT. The article is concerned with learning link grammars in the model of Gold. We show that rigid and k-valued link grammars are learnable from strings. In fact, we prove that the languages of link structured lists of words associated to rigid link grammars have finite elasticity and we show a learning algorithm. As a standard corollary, this result leads to the learnability of rigid or k-valued link grammars learned from strings.

1.1 Introduction

Link grammars, introduced in Sleator and Temperley (1991, 1993), are a formal grammatical system for natural language processing. A sentence is recognized if there is a way to connect correctly the specification of the words in the lexicon using *links*. Very close to dependency grammars Mel'cuk (1988), link grammars have the advantage to show directly the connections of words in a graph (not only as a tree like dependency grammars) which can be useful for a direct translation from syntax to semantics.

Since link grammars are completely lexicalized, they are well adapted to learning perspectives and an actual way of research is to determine what are the lexicalized grammars that remain learnable in the sense of Gold (1967). In fact, learning link grammars was mainly studied for probabilistic variants Pietra et al. (1994); Fong and Wu (1995); Kübler (1998) or with a context where only one word is unknown and has to be inferred in a sentence Pedersen and Chen (1995).

Here, we recall that learning consists to define an algorithm on a finite set of sentences that converge to obtain a grammar in the class that generates the examples. Let $\mathcal{G}$ be a class of grammars that we wish to learn from positive examples. Formally, let $\mathcal{L}(G)$ denote the language associated with grammar G, and let V be a given alphabet. A learning algorithm is a function ϕ from finite sets of words in V^* to $\mathcal{G}$, such that for $G \in \mathcal{G}$ with $\mathcal{L}(G) = (e_i)_{i\in\mathbb{N}}$ there exist a grammar $G' \in \mathcal{G}$ and $n_0 \in N$ such that: $\forall n > n_0,\ \phi(\{e_1, \ldots, e_n\}) = G' \in \mathcal{G}$ and $\mathcal{L}(G') = \mathcal{L}(G)$. After the initial pessimism following the unlearnability results in Gold (1967), there has been a renewed interest due to learnability of non trivial classes from Angluin

Proceedings of Formal Grammar 2003
G. Jäger, P. Monachesi, G. Penn & S. Wintner (editors).
Chapter 1, Copyright ©2003, Denis Béchet.

Draft

Draft

(1980) and Shinohara (1990). Recent works from Kanazawa (1998) have answered the problem for different classes associated to classical categorial grammars. In the paper, we try to answer to the question for link grammars: are they learnable from strings?

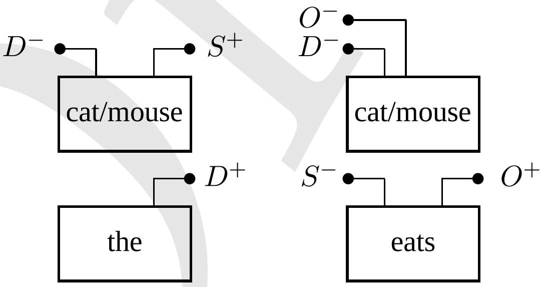
The paper is organized as follows. Section 2 presents link grammars. Section 3 reminds some useful properties on languages and learning algorithms. Section 4 gives the main lemma: rigid untyped link net grammars have finite elasticity and thus is learnable. This section ends with a learning algorithm for this class of structured languages. Section 5 extends this result and proves that k-valued link grammars are learnable from strings. Section 6 concludes and gives some perspectives.

1.2 Link Grammar

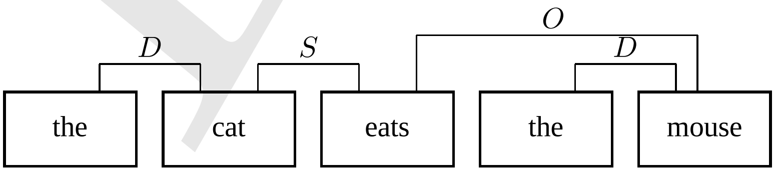
A link grammar connects the words of a sentence by links so as to satisfy:

- **Planarity**: The links do not cross.
- **Connectivity**: The graph is connected.
- **Ordering**: The lexicon gives the left and right possible linkages of each word.
- **Exclusion**: No two links may connect the same pair of words.

For instance, the link grammar with the following nodes:



accepts the sentence “the cat eats the mouse”:



A link is annotated with a primitive type that must match the left and right constraints of the two linked words. It corresponds to a functional dependency: S for subject, D for determinant, O for object.

Draft

1.2.1 Formal Definitions

- Let Σ be an *alphabet* and Pr a set of *primitive types*.
- A *link net* is a non-empty, planar and connected graph with a total order on vertices which must be on the border of the graph and where edges are annotated by primitive types and vertices by words. The set of link nets over Σ is noted $\mathcal{N}_{Pr}(\Sigma)$. Formally, let $(\mathcal{V}, \leq)$ be a totally ordered set ($\mathbb{N}$ for instance), a link net is a structure (V, w, E, t) where:
  - $V \subseteq \mathcal{V}$, the vertices, is a non-empty finite subset of $\mathcal{V}$ written $(v_1, \dots, v_n)$ where $n = \#V$ and $v_1 < \dots < v_n$;
  - $w : V \mapsto \Sigma$ maps each vertex to a word;
  - $E \subseteq V \times V$, the edges, is a symmetrical[1] and anti-reflexive[2]subset of $V \times V$;
  - $t : E \mapsto Pr$ maps each edge to a primitive type;
  - The edges do not cross[3]: if $(a, b) \in E$ and $(c, d) \in E$ such that $a < b$ and $c < d$ then it is not possible that $a < c < b < d$ or $c < a < d < b$.
  - The graph (V, E) is connected[4].

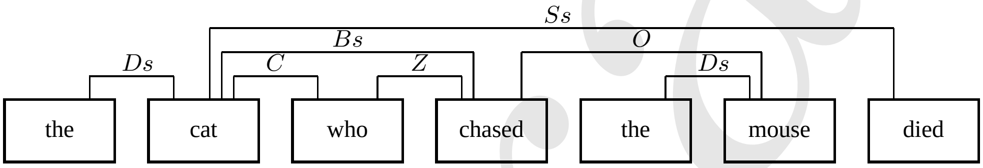


Figure 1.1: A link net

- An *untyped link net* is a link net where edges are not annotated by primitive types. If $N = (V, w, E, t)$ is a link net, we write $untyped(N) = (V, w, E)$ the corresponding untyped link net. The set of untyped link nets over Σ is noted $\mathcal{N}(\Sigma)$.
- The *yield* of a link net $N = ((v_1, \dots, v_n), w, E, t)$ or of an untyped link net $N = ((v_1, \dots, v_n), w, E)$ is $yield(N) = w(v_1) \cdots w(v_n) \in \Sigma^+$.
- The set of *link nodes* over Pr noted Tp is the set of pairs of finite lists of Pr. A link node X has a *left list of ports* noted $t_n^-, \dots, t_1^-$ and a *right list of ports* noted $t_1^+, \dots, t_m^+$. The *left arity* noted $a^l(X)$ is n and the *right arity* noted $a^r(X)$ is m. A link node is noted by its ports $t_n^-, \dots, t_1^-, t_1^+, \dots, t_m^+$. An *untyped link node* is just characterized by its left and right arity.
- For each vertex v of a link net $N = (V, w, E, t)$, the set of edges ending in v can be split into a left and a right lists $(x_n, v), \dots, (x_1, v)$ and $(v, y_1), \dots, (v, y_m)$ where $x_n < x_{n-1} < \dots < x_1 < v < y_1 < \dots < y_{m-1} < y_m$. Thus, v is associated to the link node $node(v) = t(x_n, v)^-, \dots, t(x_1, v)^- t(v, y_1)^+, \dots, t(v, y_m)^+$.

[1] A relation E is symmetrical iff $(x, y) \in E \Leftrightarrow (y, x) \in E$

[2] A relation E is anti-reflexive iff $(x, x) \notin E$

[3] Thus the graph (V, E) is planar and the vertices are on its border.

[4] i.e. for every couple of vertices, there exists a path between them.

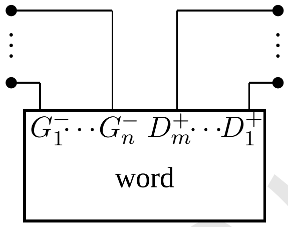


Figure 1.2: A link node

- A *link grammar* is a structure $G = (\Sigma, I)$ where $I : \Sigma \mapsto \mathcal{P}^f(Tp)$ is a function that maps to each element of Σ a finite set of *link nodes*.
- For $a \in \Sigma$, if $A \in I(a)$, G *associates* A to a (written $G : a \mapsto A$).
- A link net $((v_1, \ldots, v_n), w, E, t)$ is generated by G iff $G : w(v_i) \mapsto t(v_i)$ for all i, $0 \leq i \leq n$.
- An untyped link net N is generated by G iff it exists a link net N' such that $N = untyped(N')$ and N' is generated by G.
- A sentence $c_1 \cdots c_n \in \Sigma^+$ is generated by G iff it exists a link net N such that $c_1 \cdots c_n = yield(N)$ and N is generated by G.
- The language of link nets of G, noted $\mathcal{L}_{\mathcal{N}_{Pr}(\Sigma)}(G)$ is the set of link nets generated by G.
- The language of untyped link nets of G, noted $\mathcal{L}_{\mathcal{N}(\Sigma)}(G)$ is the set of untyped link nets generated by G.
- The language of strings of G, noted $\mathcal{L}_{\Sigma^+}(G)$ is the set of sentences generated by G.
- Link grammars that associate at most k link nodes to each symbol of Σ are called k-*valued*.
- 1-valued grammars are also called *rigid*.

1.3 Learnability and Finite Elasticity

We have seen in the introduction that a class of languages described by a class of grammars $\mathcal{G}$ is learnable iff there exists a learning algorithm ϕ from finite sets of words to $\mathcal{G}$ that converges to G[5] for any $G \in \mathcal{G}$ and for any growing partial enumeration of $\mathcal{L}(G)$.

Learnability and unlearnability properties have been widely studied from a theoretical point of view. A very useful property for our purpose is the finite elasticity property of a class of languages. This term was first introduced in Wright (1989); Motoki et al. (1991) and, in fact, it induces learnability. A very nice presentation of this notion can be found in Kanazawa (1998).

[5] In fact, it is not the output grammars that converge but their associated languages.

Draft

Draft

Definition

- A class $\mathcal{CL}$ of languages has *infinite elasticity* iff $\exists (e_i)_{i \in \mathbb{N}}$ an infinite sequence of sentences, $\exists (L_i)_{i \in \mathbb{N}}$ an infinite sequence of languages of $\mathcal{CL}$ such that $\forall i \in \mathbb{N} \ : \ e_i \not\in L_i$ and $\{e_0, \ldots, e_{i-1}\} \subseteq L_i$.
- A class has *finite elasticity* iff it has not infinite elasticity.

Theorem 1 [Wright 1989] *A class that is not learnable has infinite elasticity.*

Corollary 2 *A class that has finite elasticity is learnable.*

Finite elasticity is a very nice property because it can be extended from a class to another one that is obtained by a *finite-valued relation*[6]. We use here a version of the theorem that has been proved in Kanazawa (1998) and is useful for various kind of languages (strings, structures, nets) that can be described by lists of elements of some alphabets.

Theorem 3 [Kanasawa 1998] *Let $\mathcal{M}$ be a class of languages over Γ that has finite elasticity, and let $R \subseteq \Sigma^* \times \Gamma^*$ be a finite-valued relation. Then the class of languages $\{R^{-1}[M] = \{s \in \Sigma^* \mid \exists u \in M \wedge (s, u) \in R\} \mid M \in \mathcal{M}\}$ has finite elasticity.*

1.4 Finite Elasticity of Rigid Untyped Link Net Grammars

This section is concerned by grammars of untyped link nets rather than grammars of strings. The following theorem is essential because, as a corollary, the corresponding class of rigid link grammars has finite elasticity and thus is learnable from strings. This result can also be extended to the class of k-valued link grammars for every k.

Theorem 4 *Rigid link grammars define a class of languages of untyped link nets that has finite elasticity.*

Proof: We use a result of Shinohara Shinohara (1990, 1991) that proves that *formal systems* that has *finite thickness* has finite elasticity. In Shinohara (1991) this is applied to *length-bounded elementary formal system with at most k rules* and also to *context sensitive languages* that are definable by at most k rules. Formal systems in Shinohara (1991) do not describe languages of strings only but also languages of terms. It can be applied to typed or untyped link nets which can be seen as well-bracketed strings (each link is associated to an opening and a closing (typed) bracket). For the class of rigid untyped link net grammars, the sketch of proof is as follows:

[6] A relation $R \subseteq \Sigma^* \times \Gamma^*$ is finite-valued iff for every $s \in \Sigma^*$, there are at most finitely many $u \in \Gamma^*$ such that $(s, u) \in R$.

Draft

1. **Definition.** A link grammar $G_1 = (\Sigma_1, I_1)$ is *included* in a link grammar $G_2 = (\Sigma_2, I_2)$ (notation $G_1 \subseteq G_2$) iff $\Sigma_1 \subseteq \Sigma_2$ and $\forall x \in \Sigma_1$, $I_1(x) \subseteq I_2(x)$.

2. **Definition and lemma.** The mapping $\mathcal{L}_{\mathcal{N}(\Sigma)}$ from link grammars to untyped link net languages is monotonic: if $G_1 \subseteq G_2$ then $\mathcal{L}_{\mathcal{N}(\Sigma)}(G_1) \subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(G_2)$.

3. **Definition.** A grammar G is *reduced with respect to a set* $X \subseteq \mathcal{N}(\Sigma)$ iff $X \subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(G)$ and for each grammar $G' \subseteq G$, $X \not\subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(G')$. Intuitively, a grammar that is reduced with respect to X, does not have redundant expression to cover all the structures of X.

4. **Lemma.** For each finite set $X \subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(G)$, there is a finite set of rigid untyped link net languages that correspond to the grammars that are reduced from X. This is the main part of the proof. In fact, if $G = (\Sigma, I)$, a rigid untyped link net grammar, is reduced with respect to X then each word that does not appear in one of the untyped link net of X must be associated through I to the empty set. The other words must be associated to exactly one type of Tp (the grammar is rigid). The left and the right arities are given by the occurrences of the word in the untyped link nets and they must be the same for all the occurrences because the language we try to learn corresponds to a *rigid* untyped link net grammar. If the sum of the left and right arities of each word in X is bound by m, and if n is the number of words that appear in X, the number of equivalent grammars[7] is bound by the number of partitions of a set of $n \times m$ elements.

5. **Definition.** Monotonicity and the previous property define a system that has *bounded finite thickness*.

6. **Theorem.** Shinohara proves in Shinohara (1991) that a formal system that has bounded finite thickness has finite elasticity.

7. **Corollary.** Rigid untyped link net languages have finite elasticity.

A learning algorithm for rigid untyped link net grammars

In fact, the notion of reduced grammars suggests a simple learning algorithm for the class of rigid untyped link net grammars. The algorithm ϕ_1 takes a sequence of untyped link nets $N_1, \ldots, N_l$ and produces a link grammar that corresponds to the smallest rigid untyped link net language that is compatible to the input. Of course, this algorithm returns a failure if the sequence does not correspond to a rigid untyped link net language[8].

[7] Equivalent grammars are grammars that are associated to the same language. A sufficient condition is the existence of a bijective relation between the primitive types of both grammars.

[8] i.e. if this sequence is not included in at least one of the rigid untyped link net languages.

Draft

1. We collect the occurrences of each word together with its left and right arities from the input sequence. The algorithm fails if a word is used with different left or right arities.

2. For each word w, we associates $n + m$ variables corresponding to the n left ports and the m right ports: $X^w_{-n}, \dots, X^w_{-1}, X^w_1, \dots, X^w_m$

3. Then, we extract equality constraints on the variables based on the links that appear in the input untyped link nets: a link corresponds to two variables on both ends that must be equal.

4. The equality system is resolved and a primitive type is associated to each equivalence class of variables noted $\overline{X^w_i}$.

5. For each word w that appears in the sequence and is associated to the $n+m$ variables $X^w_{-n}, \dots, X^w_{-1}, X^w_1, \dots, X^w_m$, the output link grammar associates the link node $\overline{X^w_{-n}}^-, \dots, \overline{X^w_{-1}}^-, \overline{X^w_1}^+, \dots, \overline{X^w_m}^+$ to w.

Theorem 5 *ϕ_1 learns rigid untyped link net grammars.*

Proof: ϕ_1 is monotonic: if $S_1 \subseteq S_2$ then $\phi_1(S_2)$ returns a failure or $\phi_1(S_1)$ and $\phi_1(S_2)$ succeed and $\mathcal{L}_{\mathcal{N}(\Sigma)}(\phi_1(S_1)) \subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(\phi_1(S_2))$. This is a consequence of the fact that the equality system corresponding to S_1 is a subset of the equality system corresponding to S_2.

Let $G = (\Sigma, I)$ be a rigid link grammar and $(N_i)_{i\in\mathbb{N}}$ an infinite sequence of untyped link nets that enumerates $\mathcal{L}_{\mathcal{N}(\Sigma)}(G)$. For $i \in \mathbb{N}$, $\phi_1(N_0, \dots, N_i)$ does not return a failure because G is rigid so each word in the untyped link nets of $\mathcal{L}_{\mathcal{N}(\Sigma)}(G)$ is used with the same left and right arities. Because ϕ_1 is monotonic, $(G_i = \phi_1(N_0, \dots, N_i))_{i\in\mathbb{N}}$ defines an infinite sequence of growing languages $\mathcal{L}_{\mathcal{N}(\Sigma)}(G_0) \subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(G_1) \subseteq \cdots$. Because the class has finite elasticity, this sequence must converge to a language L_∞ that must be (equal or) a superset of $\mathcal{L}_{\mathcal{N}(\Sigma)}(G)$ since the sequence enumerates $\mathcal{L}_{\mathcal{N}(\Sigma)}(G)$. In fact, for $i \in \mathbb{N}$, G verifies the equality system used by ϕ_1 with $N_0, \dots, N_i$ as input, so $\mathcal{L}_{\mathcal{N}(\Sigma)}(\phi_1(N_0, \dots, N_i)) \subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(G)$. Thus, we also have $L_\infty \subseteq \mathcal{L}_{\mathcal{N}(\Sigma)}(G)$. ■

1.5 k-valued Link grammars are learnable from strings

Because the class of rigid untyped link net languages has finite elasticity, we can find a finite-valued relation between rigid untyped link net languages and k-valued link languages. In fact, we define two relations and use twice Theorem 3. The first one from rigid untyped link net languages to rigid link languages and the second from rigid link languages to k-valued link languages.

Lemma 6 *Rigid link languages have finite elasticity.*

Draft

Proof: An untyped link net $((v_1, \ldots, v_n), w, E)$ over Σ is characterized by the left and right arities of each vertex. Thus, it can be completely described by the following string from the alphabet $\Sigma \cup \{\mathbf{[}\,,\mathbf{]}\}$ where **[** and **]** are not in Σ:

$$w(v_1)\,\underbrace{\mathbf{[}\cdots\mathbf{[}}_{a^r(v_1)}\,\underbrace{\mathbf{]}\cdots\mathbf{]}}_{a^l(v_2)}\,w(v_2)\,\underbrace{\mathbf{[}\cdots\mathbf{[}}_{a^r(v_2)}\;\cdots\;\underbrace{\mathbf{]}\cdots\mathbf{]}}_{a^l(v_{n-1})}\,w(v_{n-1})\,\underbrace{\mathbf{[}\cdots\mathbf{[}}_{a^r(v_{n-1})}\,\underbrace{\mathbf{]}\cdots\mathbf{]}}_{a^l(v_n)}\,w(v_n)$$

In fact, the relation between Σ^+ and $(\Sigma \cup \{\mathbf{[}\,,\mathbf{]}\})^+$ that adds square brackets such that the result corresponds to an untyped link net is finite-valued. Then, because the class of languages of $(\Sigma \cup \{[,]\})^+$ corresponding to rigid untyped link net grammars has finite elasticity, their inverse images by the previous relation, which define exactly the class of rigid link languages, have finite elasticity. ■

Lemma 7 *k-valued link languages have finite elasticity.*

Proof: This is very standard. The finite-valued relation associates k-valued link grammars over Σ and rigid link grammars over $\Sigma \times \{1, \ldots, k\}$. A rigid link grammar over $\Sigma \times \{1, \ldots, k\}$ corresponds to the k-valued link grammar where the types associated to $(a, 1), \ldots, (a, k)$ are merged into the same entry for a. ■

1.6 Conclusion and perspectives

We have proved in the paper that the class of rigid untyped link net languages has finite elasticity. We have given a learning algorithm for this class of languages of untyped link nets. Finally, we have proved that k-valued link grammars have also finite elasticity and thus is learnable from strings.

This positive result may be compared to other learnability results in the same domain in particular in the field of k-valued categorial grammars. Kanazawa's positive result on classical categorial grammar is very similar to our result but other more complex but very close systems like Lambek calculus or pregroups have been proved to be not learnable from strings Foret and Le Nir (2002a,b). Due to the similarity between link grammars and Lambek calculus or pregroups presented as (proof) nets and their differences from the learnability point of view, we can try to deduce general rules for learnable classes.

For us, the reason why k-valued Lambek calculus grammars are not learnable comes from the fact that using a sequence of sentences, we can not bound the possible "interactions" (in term of proof nets, it means different set of axiom links) between two words which is not the case for k-valued link grammars because for one type there is only one possible interaction between two words. This remark may be interesting for defining useful learnable classes of logical categorial grammars that lay between classical categorial grammars and Lambek calculus.

Bibliography

Angluin, D. (1980). Inductive inference of formal languages from positive data. *Information and Control*, **45**:117–135.

Fong, E. and D. Wu (1995). Learning restricted probabilistic link grammars. In *IJCAII Workshop on New Approaches to Learning for Natural Language Processing*, pp. 49–56. Montreal, Canada.

Foret, A. and Y. Le Nir (2002a). Lambek rigid grammars are not learnable from strings. In *COLING'2002, 19th International Conference on Computational Linguistics*. Taipei, Taiwan.

Foret, A. and Y. Le Nir (2002b). On limit points for some variants of rigid lambek grammars. In *ICGI'2002, the 6th International Colloquium on Grammatical Inference*, number 2484 in Lecture Notes in Artificial Intelligence. Springer-Verlag.

Gold, E. (1967). Language identification in the limit. *Information and control*, **10**:447–474.

Kanazawa, M. (1998). *Learnable classes of categorial grammars*. Studies in Logic, Language and Information. FoLLI & CSLI. Distributed by Cambridge University Press.

Kübler, S. (1998). Learning a lexicalized grammar for german. In D. Powers, ed., *NeMPaP3/CoNLL98: New Methods in Language Processing and Computational Natural Language Learning*, pp. 11–18. ACL.

Mel'cuk, I. (1988). *Dependency Syntax: Theory and Practive*. State University of New York Press.

Motoki, T., T. Shinohara, and K. Wright (1991). The correct definition of finite elasticity: Corrigendum to identification of unions. In *The fourth Annual Workshop on Computational Learning Theory*, p. 375. Morgan Kaufmann, San Mateo, Calif.

Pedersen, T. and W. Chen (1995). Lexical acquisition via constraint solving. In *Working Notes of the AAAI Spring Symposium on Representation and Acquisition of Lexical Knowledge*, pp. 118–122.

Pietra, S. D., V. D. Pietra, J. Gillett, J. Lafferty, H. Printz, and L. Ures (1994). Inference and estimation of a long-range trigram model. In *Grammatical Inference and Applications: Proceedings of the Second International Colloquim ICGI-94*, pp. 78–92. Alicante, Spain.

Shinohara, T. (1990). Inductive inference from positive data is powerful. In *The 1990 Workshop on Computational Learning Theory*, pp. 97–110. Morgan Kaufmann, San Mateo, California.

Draft

Shinohara, T. (1991). Inductive inference of monotonic formal systems from positive data. *New Generation Computing*, **8 (4)**:371–384. Special Issue on Algorithmic Learning Theory for ALT'90.

Sleator, D. and D. Temperley (1991). Parsing English with a link grammar. Technical Report Tech Rep. CMU-CS-91-196, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA.

Sleator, D. and D. Temperley (1993). Parsing English with a link grammar. In *Third International Workshop on Parsing Technologies*.

Wright, K. (1989). Identifications of unions of languages drawn from an identifiable class. In *The 1989 Workshop on Computational Learning Theory*, pp. 328–333. Morgan Kaufmann, San Mateo, Calif.

Draft