Parsing English with a Link Grammar

Daniel D. Sleator* and Davy Temperley†

*School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
email: sleator@cs.cmu.edu

†Music Department
Columbia University
New York, NY 10027
email: dt3@cunixa.cc.columbia.edu

Abstract

We define a new formal grammatical system called a *link grammar*. A sequence of words is in the language of a link grammar if there is a way to draw *links* between words in such a way that (1) the local requirements of each word are satisfied, (2) the links do not cross, and (3) the words form a connected graph. We have encoded English grammar into such a system, and written a program (based on new algorithms) for efficiently parsing with a link grammar. The formalism is lexical and makes no explicit use of constituents and categories. The breadth of English phenomena that our system handles is quite large. A number of sophisticated and new techniques were used to allow efficient parsing of this very complex grammar. Our program is written in C, and the entire system may be obtained via anonymous ftp. Several other researchers have begun to use link grammars in their own research.

1 Introduction

Most sentences of most natural languages have the property that if arcs are drawn connecting each pair of words that relate to each other, then the arcs will not cross [10, p. 36]. This well-known phenomenon, which we call *planarity*, is the basis of *link grammars* our new formal language system.

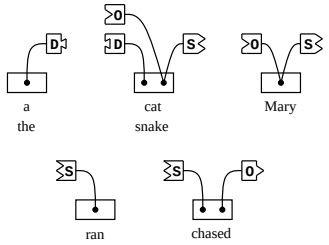
A link grammar consists of a set of *words* (the terminal symbols of the grammar), each of which has a *linking requirement*. A sequence of words is a *sentence* of the language defined by the grammar if there exists a way to draw *links* among the words so as to satisfy the following conditions:

Planarity: The links do not cross (when drawn above the words).

Connectivity: The links suffice to connect all the words of the sequence together.

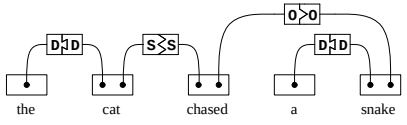
Satisfaction: The links satisfy the linking requirements of each word in the sequence.

The linking requirements of each word are contained in a *dictionary*. To illustrate the linking requirements, the following diagram shows a simple dictionary for the words *a*, *the*, *cat*, *snake*, *Mary*, *ran*, and *chased*. The linking requirement of each word is represented by the diagram above the word.

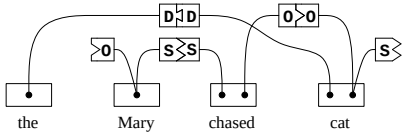


Each of the intricately shaped labeled boxes is a *connector*. A connector is satisfied by matching it to a compatible connector (one with the appropriate shape, facing in the opposite direction). Exactly one of the connectors attached to a given black dot must be satisfied (the others, if any, must not be used). Thus, *cat* requires a `D` connector to its left, and either an `O` connector to its left or a `S` connector to its right. Plugging a pair of connectors together corresponds to drawing a link between that pair of words.

The following diagram shows how the linking requirements are satisfied in the sentence *The cat chased a snake*.

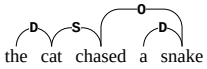


(The unused connectors have been suppressed here.) It is easy to see that *Mary chased the cat*, and *the cat ran* are also sentences of this grammar. The sequence of words: *the Mary chased cat* is not in this language. Any attempt to satisfy the linking requirements leads to a violation of one of the three rules. Here is one attempt:



Similarly *ran Mary*, and *cat ran chased* are not part of this language.

A set of links that prove that a sequence of words is in the language of a link grammar is called a *linkage*. From now on we'll use simpler diagrams to illustrate linkages. Here is the simplified form of the diagram showing that *the cat chased a snake* is part of this language:



We have a succinct, computer-readable notation for expressing the dictionary of linking requirements. The following dictionary encodes the linking requirements of the previous example.

words	formula
a the	`D+`
snake cat	`D- & (O- or S+)`
Mary	`O- or S+`
ran	`S-`
chased	`S- & O+`

The linking requirement for each word is expressed as a formula involving the operators `&`, and `or`, parentheses, and connector names. The `+` or `-` suffix on a connector name indicates the direction (relative to the word being defined) in which the matching connector (if any) must lie. The `&` of two formulas is satisfied by satisfying both the formulas. The `or` of two formulas requires that exactly one of its formulas be satisfied. The order of the arguments of an `&` operator is significant. The farther left a connector is in the expression, the nearer the word to which it connects must be. Thus, when using *cat* as an object, its determiner (to which it is connected with its `D-` connector) must be closer than the verb (to which it is connected with its `O-` connector).

We can roughly divide our work on link grammars into three parts: the link grammar formalism and its properties, the construction of a wide-coverage link grammar for English, and efficient algorithms and techniques for parsing link grammars. We now touch briefly on all three of these aspects.

Link grammars are a new and elegant context-

free grammatical formalism[1][2], and have a unique combination of useful properties:

1. In a link grammar each word of the lexicon is given a definition describing how it can be used in a sentence. The grammar is distributed among the words. Such a system is said to be *lexical*. This has several important advantages. It makes it easier to construct a large grammar, because a change in the definition of a word only affects the grammaticality of sentences involving that word. The grammar can easily be constructed incrementally. Furthermore, expressing the grammar of the irregular verbs of English is easy – there's a separate definition for each word.

   Another nice feature of a lexical system is that it allows the construction of useful probabilistic language models. This has led researchers to construct lexical versions of other grammatical systems, such as tree-adjoining grammars [13]. Lafferty and the present authors have also constructed such a probabilistic model for link grammars [11].

2. Unlike a phrase structure grammar, after parsing a sentence with a link grammar words that are associated semantically and syntactically are directly linked. This makes it easy to enforce agreement, and to gather statistical information about the relationships between words.

3. In English, whether or not a noun needs a determiner is independent of whether it is used as a subject, an object, or even if it is part of a prepositional phrase. The algebraic notation we developed for expressing a link grammar takes advantage of this orthogonality. Any lexical grammatical system, if it is to be used by a human, must have such a capability. In our current on-line dictionary the word *cat* can be used in 369 different ways, and for *time* this number is 1689. A compact link grammar formula captures this large number of possibilities, and can easily be written and comprehended by a human being.

4. Another interesting property of link grammars is that they have no explicit notion of constituents or categories. In most sentences parsed with our dictionaries, constituents can be seen to emerge as contiguous connected collections of words attached to the rest of the sentence by a particular type of link. For example in the dictionary above, `S` links always attach a noun phrase (the connected collection of words at the left end of the link) to a verb (on the right end of the link). `O` links work in a similar fashion. In these cases the links of a sentence can be viewed as an alternative way of specifying the constituent structure of the sentence. On the other hand, this is not the way we think about link grammars, and we see no advantage in taking that perspective.

Our second result is the construction of a link grammar dictionary for English. The goal we set for ourselves was to make a link grammar that can distinguish, as accurately as possible, syntactically correct English sentences from incorrect ones. We chose a formal or newspaper-style English as our model. The result is a link grammar of roughly eight hundred definitions (formulas) and 25000 words that captures many phenomena of English grammar. It handles: noun-verb agreement, questions, imperatives, complex and irregular verbs, many types of nouns, past- or present-participles in noun phrases, commas, a variety of adjective types, prepositions, adverbs, relative clauses, possessives, coordinating conjunctions, unbounded dependencies, and many other things.

The third result described in this paper is a program for parsing with link grammars. The program reads in a dictionary (in a form very similar to the tables above) and will parse sentences according to the given grammar. The program does an exhaustive search – it finds every way of parsing the given sequence with the given link grammar. It is based on our own $O(n^3)$ algorithm (n is the number of words in the sentence to be parsed). The program also makes use of several very effective data structures and heuristics to speed up parsing. The program is comfortably fast – parsing typical newspaper sentences in a few seconds on a modern workstation.

Both our program (written in ANSI-C) and our dictionary are available via anonymous ftp through the internet.[3] Having the program available for experimentation may make it easier to understand this paper.

[1] Link grammars resemble dependency grammars and categorial grammars. There are also many significant differences. Some light is shed on the relationship in section 6.

[2] The proof of the context-freeness of link grammars is not included in this paper, but appears in our technical report [14]. Note that context-free systems can differ in many ways, including the ease with which the same grammar can be expressed, the efficiency with which the same grammar can be parsed, and the usefulness of the output of the parser for further processing.

The organization of this paper

In section 2 we define link grammars more formally and explain the notation and terminology used throughout the rest of the paper. In section 3 we describe the workings of a small link grammar for English. Our $O(n^3)$ algorithm is described in section 4, and the data structures and heuristics that make it run fast are described in section 5. In section 6 we explain the relationship between link grammars, dependency syntax, and categorial grammars. We show how to automatically construct a link grammar for a given categorial grammar. This construction allows our efficient parsing algorithms and heuristics to be applied to categorial grammars. Section 7 mentions several other research projects that are based on link grammars.

Space limitations prevent us from presenting details of a number of other aspects of our work. The following paragraphs mention a few of these. More details on all of these matters are contained in our technical report [14].

There are a number of common English phenomena that are not handled by our current system. Our technical report contains a list of these, along with the reason for this state of affairs. The reasons range from the fact that ours is a preliminary system to the fact that some phenomena simply do not fit well into the link grammar framework.

Coordinating conjunctions such as *and* pose a problem for link grammars. This is because in a sentence like *The dog chased and bit Mary* there should logically be links between both *dog* and *bit* and *chased* and *Mary*. Such links would cross. We have devised a scheme that handles the vast majority of uses of such conjunctions and incorporated it into our program. The existence of such a conjunction in a sentence modifies the grammar of the words in it. The same parsing algorithm is then used on the resulting modified grammar.

Certain other constructs are difficult to handle only using the basic link grammar framework. One example is the non-referential use of *it*: *It is likely that John will go* is correct, but *The cat is likely that John will go* is wrong. It is possible – but awkward – to distinguish between these with a link grammar. To deal with this (and a number of other phenomena), we extended the basic link grammar formalism with a *post-processor* that begins with a linkage, analyzes its structure, and determines if certain conditions are satisfied. This allows the system to correctly judge a number of subtle distinctions (including that mentioned here).

[3] The directory is `/usr/sleator/public` on the host `spade.pc.cs.cmu.edu` (128.2.209.226). Our technical reports [14, 11] are also available there.

2 Notation and terminology

2.1 Meta-rules

The link grammar dictionary consists of a collection of entries, each of which defines the linking requirements of one or more words. These requirements are specified by means of a *formula* of *connectors* combined by the binary associative operators `&` and `or`. Presidence is specified by means of parentheses. Without loss of generality we may assume that a connector is simply a character string ending in `+` or `-`.

When a link connects to a word, it is associated with one of the connectors of the formula of that word, and it is said to *satisfy* that connector. No two links may satisfy the same connector. The connectors at opposite ends of a link must have names that *match*, and the one on the left must end in `+` and the one on the right must end in `-`. In basic link grammars, two connectors match if and only if their strings are the same (up to but not including the final `+` or `-`). A more general form of matching will be introduced later.

The connectors satisfied by the links must serve to satisfy the whole formula. We define the notion of satisfying a formula recursively. To satisfy the `&` of two formulas, both formulas must be satisfied. To satisfy the `or` of two formulas, one of the formulas must be satisfied, and *no* connectors of the other formula may be satisfied. It is

sometimes convenient to use the empty formula ("()"), which is satisfied by being connected to no links.

A sequence of words is a *sentence* of the language defined by the grammar if there exists a way to draw links among the words so as to satisfy each word's formula, and the following *meta-rules*:

Planarity: The links are drawn above the sentence and do not cross.

Connectivity: The links suffice to connect all the words of the sequence together.

Ordering: When the connectors of a formula are traversed from left to right, the words to which they connect proceed from near to far. In other words, consider a word, and consider two links connecting that word to words to its left. The link connecting the nearer word (the shorter link) must satisfy a connector appearing to the left (in the formula) of that of the other word. Similarly, a link to the right must satisfy a connector to the left (in the formula) of a longer link to the right.

Exclusion: No two links may connect the same pair of words.

2.2 Disjunctive form

The use of formulas to specify a link grammar dictionary is convenient for creating natural language grammars, but it is cumbersome for mathematical analysis of link grammars, and in describing algorithms for parsing link grammars. We therefore introduce a different way of expressing a link grammar called *disjunctive form*.

In disjunctive form, each word of the grammar has a set of *disjuncts* associated with it. Each disjunct corresponds to one particular way of satisfying the requirements of a word. A disjunct consists of two ordered lists of connector names: the *left list* and the *right list*. The left list contains connectors that connect to the left of the current word (those connectors end in −), and the right list contains connectors that connect to the right of the current word. A disjunct will be denoted:

$$((L_1, L_2, \ldots, L_m)\ \ (R_n, R_{n-1}, \ldots, R_1))$$

Where $L_1, L_2, \ldots, L_m$ are the connectors that must connect to the left, and $R_1, R_2, \ldots, R_n$ are connectors that must connect to the right. The number of connectors in either list may be zero. The trailing + or − may be omitted from the connector names when using disjunctive form, since the direction is implicit in the form of the disjunct.

To satisfy the linking requirements of a word, one of its disjuncts must be satisfied (and no links may attach to any other disjunct). To satisfy a disjunct all of its connectors must be satisfied by appropriate links. The words to which $L_1, L_2, \ldots$ are linked are to the left of the current word, and are monotonically increasing in distance from the current word. The words to which $R_1, R_2, \ldots$ are linked are to the right of the current word, and are monotonically increasing in distance from the current word.

It is easy to see how to translate a link grammar in disjunctive form to one in standard form. This can be done simply by rewriting each disjunct as

$$(\ L_1\ \&\ L_2\ \&\ \cdots\ \&\ L_m\ \&\ R_1\ \&\ R_2\ \&\ \cdots\ \&\ R_n\ )$$

and combining all the disjuncts together with the `or` operator to make an appropriate formula.

It is also easy to translate a formula into a set of disjuncts. This is done by enumerating all ways that the formula can be satisfied. For example, the formula:

```
(A- or ()) & D- & (B+ or ()) & (O- or S+)
```

corresponds to the following eight disjuncts:

```
   ((A,D)   (S,B))
  ((A,D,O)  (B))
   ((A,D)   (S))
  ((A,D,O)  ())
     ((D)   (S,B))
   ((D,O)   (B))
     ((D)   (S))
   ((D,O)   ())
```

2.3 Our dictionary language

To streamline the difficult process of writing the dictionary, we have incorporated several other features to the dictionary language. Examples of all of these features can be found in section 3.

It is useful to consider connector matching rules that are more powerful than simply requiring the strings of the connectors to be identical.

3.4 Questions

The SI connector is used to ask questions. A question has a WH-phrase and a SI+ connector.

The SI connector is used for subject-verb inversion questions. A question has a WH-phrase and a SI+ connector.

This allows

The S is used to connect a noun to its noun. Notice that this connector is used to connect a verb to its noun. The D is for connecting a determiner to its noun. The A connector is used to connect a noun to its noun. Notice that this connector is used to connect a noun to its noun. The D is for connecting a determiner

gram.

The situation is shown in

solution.

left[b]. We can now add this

disjuncts. Suppose it finds a w

incorrect sentences:

*The dog chased cats died

*The dog who chase cats di

*The dog who John chased

*The dog John chased cats

*The dog who chased died

The following incorrect constr

epted, but can be weeded out in p

s called a *pruning step*. *pruning*
eating the pruning step until it c
pplied.

The set of disjuncts left (after
lete) is independent of the orde
teps are applied. (The pruning op
Church-Rosser property.) We ther
rdering that can be efficiently i
would be ideal if we could achieve
or pruning that is linear in the nu
ors. The scheme we propose sat

of planar directed arcs among t
a tree. Each word (except the
arc out to exactly one other wo
pass over the root word. In a li
to a dependency structure) the
undirected, and may form cycl
notion of a root word.

Gaifman [5] was the first
formal method of expressing a
mar. He shows that his model

Mel̀čuk's definition of a dep

```
peanuts:      NP
passionately: (S/NP)/(S/NP)
```

Here is the derivation of *Harr*
passionately.

```
Harry     likes     peanuts
S/(S/NP) (S/NP)/NP    NP
          ---------------
                S/NP
          ---------------
S/
```

The proposed model gracefull
grams and grammatical constr
work. Andrew Hunt[6] has devel
of the relationship of prosody a
link grammars. He has imple
and in preliminary tests, the re
ter than with other models. T
modified our parser to detect
that Francophones make when
glish.