A Practical System for Human-Like Parsing

Christian R. Huyck [1]

Abstract. This paper describes a human-like natural language parser called Plink. It works by parsing left-to-right through a sentence and keeping a complete representation of the partially read sentence. It does this by combining a sophisticated unification-based grammar and grammar rule selection heuristics.

Plink also functions in real world applications. To do this, it must process texts that are not grammatical and does this by combining a general grammar and taking advantage of preference levels in the rule selection heuristics. It has been evaluated on two parsing metrics: Parseval [4] and a dependency based metric [14]. Plink performs well but below the state of the art.

Like humans, Plink parses in linear time, and generates one interpretation that is both syntactic and semantic. It is psycholinguistically inspired.

1 Introduction and Background

Example 1. *The horse raced past the barn.*

The human parsing mechanism is unlike standard computational parsers. When a human processes a sentence, he has a largely complete representation of the sentence up to the point he is currently reading [12]. Given Example 1, the reader at some point will fixate on the word *raced*. He will have already interpreted *The horse* as a noun phrase; while reading *raced*, he will determine that it is a verb phrase and that *The horse* is the actor and subject of *raced.* [2] Later, he will attach the prepositional phrase *past the barn* as the location of *raced*.

The correct interpretation of Natural Language (NL) is the interpretation that a human produces. It is possible to generate this interpretation differently than humans do, but in general we have no system that does this. In parsing, we have no system that performs in the range of humans for correct parses in an open domain. Therefore, the human parsing mechanism (as explored by psycholinguists) can provide evidence to improve our parsers.

Example 2.*The horse raced past the barn fell.*

Left-to-right parsing with no postponement of attachment can inform work in language engineering; it can explain why Example 2 is difficult for a human to understand. It can also explain why center embedded constructions are difficult to understand.

Virtually all NL parsing systems ignore this constraint, and give interpretations to sentences which humans do not interpret; all of the parsers described in [21] fall into this category. The Plink parser, which is described in this paper, reads left-to-right and keeps a largely complete representation of the sentence up to the point that is currently being read.

The Plink parser is a human-like parser. It is linear. It is robust. It generates exactly one interpretation. Plink is based on deterministic parsing because near-determinism is an essential feature of human-like parsers [12]. Furthermore, many parsing decisions have been based on our understanding of how the human parser works. However, Plink is not currently a complete model of human parsing.

Additionally, the Plink parser is meant to be used in current Text Engineering applications. While Plink can be used to parse just “grammatical” sentences, robust parsing techniques are readily incorporated into Plink, making it even more useful; these techniques are incorporated into the system described in this paper. To evaluate its effectiveness, Plink has been measured on current parsing metrics and performs well on these.

This paper first discusses the basic parsing mechanism. The power of the system comes from the interaction between sophisticated grammar rules and grammar rule selection heuristics, which are explained next. Parsing systems should parse robustly, and Plink’s ability to do this is described next. After this, results on two popular parsing metrics are discussed. The paper concludes with a discussion of heuristics and the future work on the system, and a discussion of the ramifications of this system.

2 Plink

Plink is a deterministic parser using a complex grammar. The Plink parsing mechanism is similar to the Parsifal [15] and Fidditch [9] parsers. Plink differs from these parsers by using a more sophisticated grammar rule selection mechanism. This mechanism can take advantage of semantics as well as syntax.

If a parsing decision can be made based on the existing constituents, it is made; if not the decision is delayed. However, the decision is only temporarily delayed, and if an ambiguity exists after more processing, a parsing decision is forced. This allows it to function in real-world domains.

While it may give interpretation to only grammatical sentences, it can easily incorporate robust parsing. The system described in this paper parses robustly.

2.1 Basic Parsing Mechanism

The Plink parsing mechanism is a simple stack based parser. Its power comes from the manner in which the grammar is written, and from grammar rule selection heuristics that inspect the partial parse up to the current word.

The basic mechanism is an LR parsing mechanism [1]. A new word is pushed onto the stack. Rules are applied to the stack and it is modified. When appropriate, the next new word is pushed onto the stack. Pushing a new word onto the stack is akin to a human focusing on the word. At this stage, work on previous constituents can be completed and work can be done on the current word.

[1] Middlesex University, London, NW4 4BT, UK. email: c.huyck@mdx.ac.uk

[2] While the psycholinguistic literature may not agree on when these decisions are made, this is a valid position with evidence to support it [12].

Plink does occasionally fail to parse a sentence. Some things may be added to guarantee a complete parse (see section 3), but this does not guarantee a correct parse. This seems to be a fatal problem until it is noted that humans sometimes fail to parse sentences. Example 2 is a garden path sentence which people can not normally parse [3].

This simple stack based parsing mechanism, when driven by a good grammar and grammar rule selection heuristics, functions much like the human parsing mechanism. The key is to keep a small number of elements on the stack. These elements represent the partial parse of the sentence up to and including the current word.

2.2 The Stack

Given a sequence of words w_1 ... w_n, a grammar consisting of a set of rules and an initially empty stack, parsing proceeds by modifying the stack as follows. If no grammar rule applies, the next word is pushed on the stack. A grammar rule may be applied to a single constituent on the stack to create a new constituent, or to combine a finite number of constituents replacing the old constituents with a new one. These replaced constituents may be on the top of the stack or a finite distance from the top.

Example 3. *The horse is racing.*

The processing of Example 3 will be described. Initially *The* is the focus of attention. One rule is selected and successfully applied which states that *The* is the start of a simple noun phrase (*SNP*), so the stack contains an *SNP*. [4] A new word *horse* becomes the focus of attention, and again one grammar rule is selected and successfully applied; this is the *SNP* → *SNP common-noun* rule. It is possible that the SNP is not yet complete as in *The horse buggy...*, so that decision is delayed.

When the next word is read, a rule is applied to change the *SNP* into an *NP*. Since the next word is a finite verb, the system knows that the *NP* is complete and will not be running into a phrase like *The horse buggy...* This leaves two elements on the stack, the NP and the present tense verb *is*. A simple verb phrase (*SVP*) is begun by *is*, but it cannot be made into a complete *VP* since a more complex *SVP* may be (and in this case is) formed. At the end of processing *is*, two elements are left on the stack.

The next word *racing* is processed. *SVP* → *SVP progressive-verb* is selected and applied. The next word *period* is added to the stack. First the *SVP* is transformed into a *VP*, then the *NP* is added as the actor and subject of the *VP*. If the next word were *today* instead of *period*, these same rules would be triggered. This is different than the standard analysis. In the standard analysis, all of the later arguments are added to the *VP* before the subject *NP* is added. This approach allows the stack to get larger than is necessary. It also seems inconsistent with human processing.

Example 4 is a different sentence with the same initial words.

Example 4. *The horse is racing today.*

When a human is attending to *today* he knows that *The horse* is the actor of *is racing*. The parsing mechanism does not enforce this constraint, instead the grammar and selection rules guide the parsing, and keep the stack small.

The idea of keeping the stack small is based on a processing limitation. When the stack is small, less information needs to be considered while selecting the next rule. Humans do not have to consider the syntactic structure of earlier unrelated clauses while making present syntactic decisions. To a great extent syntactic decisions are local. However, syntactic decisions may be influenced by certain key larger structures from earlier in the sentence. Information from these structures may be examined by the grammar selection rules.

The stack is kept small by combining elements of the stack using grammar rules. Therefore, whenever it is possible, a grammar rule is applied. However, in some cases a decision can not be made and must be delayed (eg. Example 6 below). Even after delay, some decisions can not be made; in this case the goal of keeping the stack small takes precedence and a decision is made. This balance between making a correct decision and minimizing the information considered is a limitation with which humans must also contend as is evidenced by human problems with interpreting multiple center embeddings.

2.3 Grammar Rules

Plink makes use of a unification-based grammar [20]. A UBG is slightly more powerful than a context-free grammar and facilitates the integration of semantics and syntax. Each constituent may be a sophisticated Directed Acyclic Graph (DAG) representing the entire semantics of the string it covers. The element may also contain syntactic and parsing information.

Example 5.
((1) = NP
(2) = VP
(2 head syn voice) = active
(head) = (2 head)
(head sem actor) = (1 head sem)
(pref name) = active-vp-adds-actor)

Example 5 shows a sample rule for combining the NP actor to an active VP. It is applied by unifying (combining) two DAGs, into a third DAG. The initial DAGs represent the NP and the initial VP, and the rule application results in a third VP DAG that includes the NP.

The resulting VP is not complete after the application of this rule. Extra arguments may still be added. In Example 4 this rule would be applied before *today* had been integrated into the VP. Grammar rules can combine elements before one element, as traditionally defined, is complete. This has the advantage of keeping the stack size small.

The full semantics of the processing of Example 3 would reside in the *head sem* feature. The semantics take advantage of case frames [7]. They would be *(action (race, progressive) (actor horse))*. If Example 4 were parsed the slot *(time today)* would be added.

2.4 Grammar Selection Rules

The basic form of the parsing algorithm is simple, but sophisticated processing is facilitated by the grammar selection rules. The Plink grammar selection rules are unlike general parsing principles [13] but instead are quite specific. This specificity is needed to deal with the complexity of parsing natural language. The grammar selection rules inspect the contents of the top of the stack, and based on the contents, select and order several grammar rules. These rules are then applied until one is successful and the stack is modified. When no rules are successful the next word is pushed on the stack.

[3] In other sentences, people do not initially make the correct parsing decision, but backtrack, correct the mistake, and continue on. Plink does not currently account for backtracking (see section 6).

[4] The standard grammar would wait to apply the rule *SNP* → *det common-noun*, but upon seeing a determiner, it is clear that it is part of an *SNP*. The standard grammar rule can be used in Plink, but additional rules like *SNP* → *det adj common-noun* would be needed. These would lead to more grammar rules and to the stack growing larger. Since it is known that an *SNP* is coming, it is best to convert *The* to an *SNP*. This is similar to binarized trees [18].

Grammar selection rules are inspecting very complex structures. Thus they have a great deal of information to bring to any rule selection and thus to the next parsing decision.

The basic form of a grammar selection rules is a UBG equation just like a grammar rule. The difference is that the selection rules are not applied to modify the stack; only the grammar rules, when successfully applied, modify the stack.

A grammar selection rule can be described by 1: the grammar rule selected, 2: the preference level, and 3: the contents of the top of the stack. The grammar rule selected is the name of the grammar rule that is selected by this selection rule. The name is specified in the *(pref name)* feature of the grammar rule; in Example 5, the name of the rule is *active-vp-adds-actor*.

Preference level is a value indicating how "good" the grammar rule is in the current context. All of the selection rules are compared to the stack, so several grammar rules may be selected. These grammar rules are ordered by the preference mechanism.

In the current system there are six preference levels. The preference level may be hard-coded in the selection rule, or this value can be determined at runtime based on values of stack components; see [10] for more information This preference mechanism adds a great deal of power to the system. It allows the use of specific and general rules to aid in grammar development. It also enables robust parsing (see section 3).

The selection rule describes the contents of the top of the stack by a UBG equation. This equation is unified with the top of the stack. If it is successful, the grammar rule is selected with the appropriate preference value. If the equation does not unify, the grammar rule is not selected.

This equation may contain extra constituents that the grammar rule does not use. These extra constituents are the context for the selection rule. This is how a grammar rule can be applied a finite distance into the stack. While selection is context sensitive, this does not adversely affect the efficiency of the parsing algorithm.

Example 6. *VP NP PP*

Example 6. shows a prepositional phrase attachment ambiguity. Does the PP attach to the NP or the VP? This decision may be based on one of the PP's features, the *(pref attach)* feature. If the *(pref attach)* feature has the value *to-vp*, the PP will be attached to the VP; if the value is *to-np* it will attach to the NP. Assuming that it is set correctly, the appropriate attachment can be implemented by the following selection rules.

Example 7.

rule-name VP-from-VP-Object
preference-value best
stack ((1) = VP
(2) = NP
(3) = PP
(3 Pref Attach) = to-vp)

Example 8.

rule-name NP-from-NP-PP
preference-value best
stack ((1) = NP
(2) = PP
(2 Pref Attach) = to-np)

Example 7 shows the selection rule that attaches the NP as the object of the VP. This is done when the next PP will attach to the VP. Note that the grammar rule VP-from-VP-Object does not use the PP, it is only used during selection. It is context sensitive. Example 8 shows the heuristic that attaches the PP to the NP; this only occurs if the PP's *(pref attach)* feature is *to-np*.

The context sensitivity of rule selection allows finitely delayed decisions. If a decision involving adjacent constituents c_i and c_j can not be made immediately, the next constituent, c_k may lend some disambiguating information. In Example 7 the extra *PP* shows the completion of the *NP* and licenses its attachment to the VP. Additional constituents may be used to delay decisions; in the current system the largest number of extra constituents considered is 4.

If a decision had not been made by the time these extra constituents are added, then something is wrong with the selection rules, and the stack will grow too large. While developing the rules, if the stack grows too large, the developer knows an attachment has not been made when it should have been. Thus his efforts are concentrated on solving this attachment problem (see section 5).

Full-fledged unification is used on selection rules so specific or general semantic categories can be checked. A very specific selection rule could check if the top of the stack was a PP with the semantic head **telescope** using the preposition *with*, the second element of the stack was an NP with the semantic feature **human** and the third element was the verb **see**. A more general rule could use the verb **attend**. In specific semantic contexts, specific selection rules override default assumptions. These specific rules in turn could be overriden by more specific semantic context and higher preference selection rules.

Plink is similar to left-corner parsing [18] in that it uses look ahead; while Roark's system has a look ahead of one, Plink's varies depending on the need of the grammar, and selection criterion. Roark's system also keeps a variable size beam of possible interpretations while Plink has only one. Roark's system is probabilistic. Plink could take advantage of probabilities to automatically derive grammar selection rules (see section 6).

Plink uses incremental interpretation and is similar to [17]. Milward suggests that each time a word is processed, it is completely incorporated into the sentence. Plink could do this but would need to keep a stack size of two (the word being processed, and the part of the sentence already processed). Plink needs the larger stack size to resolve ambiguity. Gapping could be handled by preference features that are used solely for parsing.

3 Robustness and Grammar Rule Levels

Humans can interpret a wide range of texts including those that are not "grammatical". In many natural language applications, all texts, even those that are ungrammatical need to be interpreted. That is, each sentence needs to be parsed even if it is not traditionally grammatical. The parser should give the correct result for a grammatical sentence and a reasonable result for an ungrammatical sentence. Through the use of preference levels and a general grammar, the Plink system can give a reasonable interpretation to even ungrammatical sentences. Furthermore, this mechanism eases grammar development by accounting for grammatical constructs that are not yet handled by the normal grammar.

The current Plink grammar and selection rules have a subset that accounts for most common grammatical structures. These selection rules only use the four highest preference values. By automatically generating grammar rules and selection rules with a lower preference the system can give an interpretation to any sentence. When the sentence is accounted for by the original subset of the grammar, then the sentence will be interpreted as it would without the automatically generated grammar. Moreover, where a portion of the ungrammatical sentence is grammatical, it will be correctly interpreted by the original grammar.

Example 9. *Sentence* $\rightarrow$ *word*+

It is quite simple to generate a grammar that interprets all sentences. One simple grammar would be Example 9. Here *word* might refer to all of the lexical categories which a system can expect. This could easily be translated into a UBG that had two rules for each lexical category. Clearly, this is not a very useful grammar. However, this basic idea can be folded into an existing grammar, by the use of levels of grammatical constructs.

Many current parsing systems function as a cascade of grammars (eg. [6]). For instance, a system might function by first transforming lexemes into simple noun and verb phrases. Next it parses by combining these simple noun and verb phrases into clauses, and finishes with a pass that combines clauses into sentences. The system just described has three levels of grammar rules. Standard X-Bar [11] grammars can be thought to have the first two levels of grammar rules. A cascade of grammars can be readily folded into one Plink grammar due to the stack based parsing mechanism, and the preference levels used in grammar rule selection.

If Example 9 can be applied at each level, then the number of rules generated, and their associated heuristics can be kept to a minimum. The current system uses three levels of grammar rules as described above. At the lowest level, constituents are divided into two groups: constituents that are part of a simple noun phrase and constituents that are part of a simple verb phrase (adjective phrases and prepositional phrases are combined with simple noun phrases). Selection rules and grammar rules are generated to account for all possible combinations. This guarantees that no lexemes (terminals) will be left in the final output. A similar set of rules guarantees that no simple phrases will be left in the output, and another set of rules guarantee that no clauses will be left in the output. Thus only complete sentences are produced.

These agrammatical [5] selection rules are given a low preference value so they only apply when a "grammatical" rule does not apply. When processing a grammatical sentence, the higher preference rules apply and the agrammatical rules are never applied. Only when the original grammar is insufficient will this new grammar be applied. Additionally, as soon as the agrammatical construct is accounted for, the system will return to normal grammatical interpretation.

4 Results

Plink has been thoroughly tested in practical systems, having been used in the fifth and seventh Message Understanding Competitions [2], [3], an Arabic Text Extraction System, and as part of the NL front end for a process modelling system. The results of a particular set of grammar and selection rules on two existing parsing measurements are presented. Plink is evaluated using the Parseval [4] evaluation on the development corpus, and on a blind portion of the Wall Street Journal portion of the Penn Tree Bank (PTB) [16]. Plink is then evaluated on a dependency based evaluation [14] on a portion of the Susanne Corpus [19].

The grammar was developed by using the first nine PTB texts, that consist of 69 sentences, as the development corpus. The goal was to maximize the brackets that Parseval used and to minimize any extra structure that Parseval does not use.

[5] Agrammatical refers to rules that are not in the standard grammar. The construct may be truly ungrammatical, or it just may not be accounted for by the standard grammar.

Table 1. System Comparison

	Training PTB	Test PTB	Susanne
Plink	99.0/94.9	70.8/74.1	61.5/66.4
Collins		88.1/88.6	
Lin			71.8/79.9

Table 1 describes the results on the two metrics. Each cell shows *recall/precision* measurements. The first column shows the result of the training data on Parseval. The recall is near optimal. The precision is quite high, though not as high as the recall; this is a result of extra structure generated that Parseval does not accept. This extra structure raises the score on the dependency-based evaluation. This indicates that Plink may perform at human levels of parsing when its rule set is complete.

The second column shows the results of Parseval on unseen texts. Collins' parser [5] has the best results on this metric. Though they are lower, Plink's results compare well with Collins' results.

The third column shows the results on the dependency-based evaluation. This evaluation is different than the Parseval evaluations. Firstly, it uses a different corpus, the Susanne corpus. Secondly, instead of being based on bracketing, the parse tree is described by a dependency structure. [14] explains more thoroughly how a dependency structure can be converted into a tree. Plink uses the same grammar as in the Parseval evaluation and compares relatively well with Lin's result using his own parser. Plink suffers in the Susanne evaluation by not building extra internal structure that is needed by the evaluation.

Both Lin's and Collins' parsers would not have performed as well on the other's metric, since the annotation system is different. For instance, the Susanne corpus tends to give more structure inside simple noun phrases.

Plink performs well on the Susanne and Parseval evaluation. The grammar, heuristics and lexicon are relatively new and it is hoped that further work will lead to even better results. It is hoped that automated heuristic generation (see section 6) will increase Plink's results beyond Colins' and Lin's. The initial results are promising. It is hoped that scores can approach those generated by Plink on the training data. Plink is not currently as effective but unlike the other two, it parses ungrammatical sentences, is linear and human-like.

5 Discussion of Selection Rules

The rules described in this paper were generated by using a three tier path. 1: Adhering to levels, build a (roughly) regular grammar. 2: Handle NP boundary problems. 3: Fix phrase combining problems using semantics.

The current set of selection and grammar rules was largely built by building a regular grammar parser while adhering to levels. Combine strings of lower level constituents to form higher level constituents. A rule that was largely followed was to avoid having lower level constituents on the stack below higher level constituents. The lowest level consisted of conversion from raw lexemes into simple noun and verb phrases. Parsing like a regular grammar incorporates the right association heuristic [13].

When a simple phrase is created it is combined with existing phrases whenever possible. Similarly, complex phrases can be combined into sentences. This tier uses the second, third and fourth highest preference levels in this process. This leaves the highest level for the exceptions handled in the lower tiers of generation, and the lowest preference levels for robust parsing.

The first tier works well with one look ahead, and the remaining tiers are to fix the exceptions to the first tier. As more sentences are added to the test corpus, modification of the rules to account for these sentences forces the developer to focus on difficult attachments. The lower two tiers are where these difficult constructs are handled using semantics and look ahead.

The second tier accounts for problems with simple phrase boundaries. Tier one combines all noun-like things into one simple NP. Unfortunately, this sometimes fails as in Example 12.

Example 12. *The boy John...*

Example 12 is a simple NP when it is in *The boy John fell.* but it is two simple NPs in *The boy John hit fell.* In most cases the first tier approach works, but delaying decisions and using semantics is necessary to fix this and other problems. Other problems occur with adverbs and particles. The highest preference levels are used here to override normal processing.

The third tier accounts for phrase combination. Experience has shown that semantic decisions are rarely needed for simple phrase creation, but it is much more commonly needed for simple phrase combination. When a syntactically identical string is parsed two different ways, then it is time for semantics. Broad semantic categories can be used to influence parsing decisions. For instance, if the NP is **human** and the VP is **ingest**, then choose active-vp-adds-actor. These rules are validated by the psychological work done by [8].

The above explanation explains how this particular grammar and selection rule set works, but the selection rule mechanism should work for a general grammar. The general mechanism works by giving general selection rules a lower priority than more specific rules. The agrammatical selection rules are very general and apply in virtually any circumstance, thus they have the lowest preference. Syntactic rules that do not refer to semantic features have a higher preference value. Semantically specific rules would have higher preference values; since there is a possibility for semantic inheritance (a semantic net is built into Plink) there may be even more specific rules with even higher preference values.

Within this scheme some grammars are more suitable to this parsing mechanism than others. Combining partial analyses as quickly as possible keeps the stack size small; a grammar which waits to combine the subject-actor of an active VP until after the VP is delaying a decision it already knows. A grammar that allowed this combination would be more suitable.

6 Future Work and Conclusion

Work to date on Plink suggests several areas of future work. These include: automatic grammar and heuristic rule acquisition based on large corpora, and modifying Plink to use back-tracking.

The rule set described in this paper is rather small. Plink should avoid problems of scalability encountered by [15] because of its use of UBG, its preference scheme, and its use of semantics. A larger grammar to handle useful phenomenon (such as wh-gapping) should not affect existing grammar rules as extra features can be used to handle these new phenomenon. The current system has six preference levels, but as the semantics get more complex, extra preference levels could be used to resolve conflicts.

There has been work in automatic grammar acquisition and this work can be applied to Plink. Automatic heuristic generation is of particular interest. Many parsing decisions (such as PP attachment) are based on semantics. Heuristics can be automatically generated based on the semantic content of the constituents involved in the attachments. These heuristics can be derived by looking at the attachment decisions that are made in large corpora and the semantics of the components involved.

Backtracking should also be considered. When should a system backtrack and to where should it backtrack? In grammatical sentences people do occasionally make the incorrect parsing decision and must backtrack. To what degree can Plink duplicate this behavior? How is this related to preference levels and the size of the stack? Another issue is the relationship between grammatical backtracking, and accounting for ungrammatical sentences.

Some sentences are grammatical but humans still can not process them. This gives us extra evidence on how to correctly parse, evidence to which Plink adheres.

Plink has elucidated and extended the work on deterministic parsers. From a Text Engineering standpoint, it is efficient, effective, and can account for ungrammatical phenomena. From a psycholinguistic standpoint Plink largely adheres to current linguistic theory, and parses roughly like humans.

REFERENCES

[1] Alfred Aho, V. R. Sethi, and J. Ullman, *Compilers: Principles, Techniques and Tools*, Addison-Wesley Publishing, 1986.

[2] ARPA, *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, Morgan Kaufmann Publishers, 1993.

[3] ARPA, *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, Morgan Kaufmann Publishers, 1998.

[4] E. Black, S. Abney, D. Flickinger, and et al., 'A procedure for quantitatively comparing the syntactic coverage of english grammars', *DARPA Speech and Natural Language Workshop*, 306–311, (1991).

[5] Michael Collins, 'Three generative, lexicalised models for statistical parsing', *In Proceedings of the Association of Computational Linguistics*, (1997).

[6] Jerry Hobbs et al., 'Sri international fastus system muc-r test results and analysis', *Proceedings of the Fourth Message Understanding Conference*, (1992).

[7] Charles Filmore, 'The case for case', in *In Universals in Linguistic Theory*, eds., Emmon Bach and Robert Harns, Holt, Rinehart and Winston Inc., (1968).

[8] M. Ford, J. Bresnan, and R. Kaplan, 'A competence-based theory of syntactic closure', in *The mental representation of grammatical relations*, ed., J. Bresnan, MIT Press, (1982).

[9] D. Hindle and M. Rooth, 'Structural ambiguity and lexical relations', *Computational Linguistics*, **19**(1), (1993).

[10] C. Huyck, *PLINK: An Intelligent Natural Language Parser*, University of Michigan technical report CSE-TR-218-94, 1994.

[11] R. Jackendoff, *X-bar Syntax: A Study of Phrase Structure*, MIT Press, 1977.

[12] M. Just and P. Carpenter, 'A theory of reading: From eye fixations to comprehension', *Psychological Review*, **87**(4), 123–154, (1980).

[13] J. P. Kimball, 'Seven principles of surface structure parsing in natural language', *Cognition*, **2**(1), 15–47, (1973).

[14] Dekang Lin, 'A dependency-based method for evaluating broad-coverage parsers', *The Fourteenth International Joint Conference on Artificial Intelligence*, 1420–1425, (1996).

[15] M. Marcus, *A Theory of Syntactic Recognition for Natural Language*, MIT Press, 1980.

[16] M. Marcus, B. Santorini, and M. Marcinkiewicz, 'Building a large annotated corpus of english: The penn treebank', *Computational Linguistics*, **19**(2), 313–330, (1993).

[17] O. Milward and R. Cooper, 'Incremental interpretation: Applications, theory and relationships to dynamic semantics', *15th International Conference of Computational Linguistics*, 748–54, (1994).

[18] B. Roark and M. Johnson, 'Efficient probabilistic top-down and left-corner parsing', *37th Meeting of the ACL*, 421–8, (1999).

[19] Geoffrey Sampson, *English for the Computer*, Oxford U. Press, 1995.

[20] Stuart M. Shieber, *An Introduction to Unification-Based Approaches to Grammar*, Center for the Study of Language an Information, 1986.

[21] M. Tomita, *Current Issues in Parsing Technology*, Norwell, 1991.