On Parsing Binary Dependency Structures Deterministically in Linear Time

Harri ARNOLA[1]
Kielikone Oy
P.O. Box 126, 00211 Helsinki, Finland
harri@kielikone.fi
and
Helsinki University of Technology
Computer Science Laboratory
Espoo, Finland

Abstract

In this paper we demonstrate that it is possible to parse dependency structures deterministically in linear time using syntactic heuristic choices. We first prove theoretically that deterministic, linear parsing of dependency structures is possible under certain conditions. We then discuss a fully implemented parser and argue that those conditions hold for at least one natural language. Empirical data demonstrates that the parsing time is indeed linear. The present quality of the parser in terms of finding **the** right dependency structure for sentences is about 85%.

Introduction

Natural language sentences have ambiguities at many levels of abstraction. Since present computational algorithms can handle only partial structures, one after another, these ambiguities cause problems for parsing. A common solution is to create alternative structures in parallel, and explore a forest of possible trees in hope that the right parse tree will appear among them. This solution for processing ambiguities in parsing creates two new problems. Which tree is the right one among many in a forest? Furthermore, in the process of creating alternative structures, the number of partial trees tends to grow exponentially or at least polynomially with the number of words of a sentence. That in turn implies similar growth in the processing time.

If a parsing algorithm were able to make confidently only the right local structural choices for a sentence, it would deterministically produce only a single, correct tree. The benefits would be obvious: there would be no search for the right tree in a forest, and the processing time could be benign. However, to our best knowledge, no one has yet been able to produce a deterministic parser for a **constituent** analysis of sentences.

A dependency theory of syntactic structure indicates syntactic relations directly between the words of a sentence (e.g., Hays, 1964; Hudson, 1976, Hellwig, 1986; Mel'chuk, 1988; Robinson, 1970; Schubert, 1986; Starosta, 1988). We have studied the parsing of dependency structures over several years (Nelimarkka et al. 1984, Jäppinen et al, 1986, Valkonen et al., 1987). In this paper we discuss the final version of our fully implemented dependency parser and show that it is possible to design a heuristic deterministic dependency parser that parses sentences in linear time. The parser chooses heuristically only one direct governor among alternatives for each word in a sentence. Such a deterministic parser runs a great potential risk that at some point a wrong choice is made and the right parse tree is missed. We demonstrate empirically that the quality of the deterministic

---

[1] Formerly Harri Jäppinen

Current address: Ganesa Oy, It. Teatterik. 1 D 22, 00100 Helsinki, Finland. E-mail: harri@kielikone.fi parser can be maintained on a satisfactory level. We first discuss deterministic parsing theoretically and then proceed to discuss the implemented parser.

1 Strings and Governments

1.1 Direct governments and governed strings

Let x be a **node** that has certain formal properties. Let $S = \{x_1, x_2, x_3, ...\}$ be a well-ordered set or a **string** of such nodes. (We do not discuss the formal properties of nodes here. Later on, when nodes are interpreted as word forms, their formal properties will be morpho-syntactic attributes.) Let R be a binary, asymmetric, and antireflexive relation between the nodes of S:

(1) $R = \{ <x_i, x_j> \mid x_i, x_j \in S, x_i\ R\ x_j, \text{and } i \# j \}$

We say that x_i **directly governs** x_j or is a **direct governor** or a **regent** of x_j. Correspondingly, x_j is **directly governed** by or a **dependant** of x_i. Graphic representation indicates direct governments by arrows (Figure 1).

Rather than using just one direct government relation we admit several annotated binary relations, distinguished with integer subscripts. Let $\mathbf{R} = \{R_1, R_2, R_3, ...\}$ be a set of such binary relations.

We stipulate the following **tree constraint** for the direct government: a directly governed node has a unique direct governor.

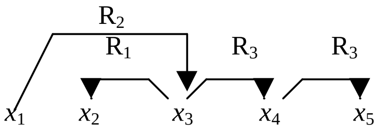

Figure 1: A governed string

We say that a node x_i **governs** x_j (i#j) iff either x_i directly governs x_j, or x_i governs x_k (k#i,j) and x_k governs or directly governs x_j. If all nodes of a string except one are governed by the same governor, we say that the string is **totally governed** (by that common governor). Figure 1 shows a totally governed string that is governed by node x_1.

Due to the tree constraint, governed strings are topologically trees. We distinguish different kinds of ambiguities in strings. A string S is **unambiguous** with respect to a set of relations **R** if each node has only one possible direct governor and governing relation. S is **locally ambiguous but globally unambiguous,** if S has only one unique totally governed string but at least one node has more than one possible direct governor or governing relation. S is (globally) **ambiguous**, if there are more than one topologically different (or differently annotated) totally governed strings for it.

We stipulate another topological constraint. The **projectivity constraint** states that if $x_i\ R_k\ x_j$, for any i, j, k and i>j, there exists no R_p such that $x_m\ R_p\ x_n$ for any m and n such that m>j and i<n<j or m<i and i<n<j. (The projectivity constraint prohibits "crossing" direct governments.)

1.2 Government Maps

It is convenient to study governed strings using two-dimensional **government maps** (GM). A GM(S,**R**) is a matrix whose rows represent the nodes of a string S and the columns represent the relations of **R**. The ordering of the rows corresponds to the ordering of the nodes in S, while the ordering of the columns (relations) is arbitrary. The direct governor of a node is marked at the intersection of the governing relation and the governed node. For example, if $\mathbf{R} = \{R_1, R_2, R_3, R_4\}$ and $S = \{x_1, x_2, x_3, x_4, x_5\}$ Table 1 shows the GM(S,**R**) of the governed string in Figure 1. Formally, GM(S,**R**) $\subset$ S x **R** x S. (Henceforth we often simply write GM rather than GM(S,**R**).)

Node/Relation	R_1	R_2	R_3	R_4
x_1				
x_2	x_3			
x_3		x_1		
x_4			x_3	
x_5			x_4	

Table 1: The GM of Figure 2

Two GM’s are called **equidimensional** if they represent identical strings and identical sets of relations and the relations occupy the same columns in both maps.

We borrow a few operations from the set theory. A direct government x_i R$_k$ x_j **belongs** to a GM (marked $\in$) iff x_i and x_j are nodes in the GM and x_i R$_k$ x_j is marked in GM. A government map GM1 **includes** another equidimensional map GM2 (GM2 $\subseteq$ GM1) if all direct governors in GM2 are also in GM1. GM1 **properly includes** an equidimensional map GM2 (GM2 $\subset$ GM1) if GM2 $\subseteq$ GM1 and GM1 $\neq$ GM2. Any given two equidimensional maps are identical, if they include one another. We also admit **unions** ($\cup$) and **intersections** ($\cap$) of two equidimensional GM’s in the obvious manner.

A GM may exhibit just those direct governers which constitute a totally governed string, it may show any subset of the direct governors of the nodes, or it may exhibit all possible direct governors of the nodes. We say that a **resolved** GM (GM^{r}) is a map that shows only the direct governors of a totally governed string. A **complete and unresolved** GM (GM^{cu}) is a map that indicates all possible direct governors of the nodes. A (partially) **unresolved** GM (GM^{u}) indicates some but not necessarily all direct governors of the nodes. For each GM, GM^{r} $\subseteq$ GM^{cu} and GM^{u} $\subseteq$ GM^{cu}.

Let GM^{r}(S,**R**) and GM^{cu}(S,**R**) represent a resolved and the complete and unresolved equidimensional maps, respectively. If S is unambiguous, GM^{r} = GM^{cu}. If S is locally ambiguous but globally unambiguous, there exists only one GM^{r} and the GM^{r} $\subset$ GM^{cu}. If S is globally ambiguous, there exists more than one different GM^{r} and for each GM^{r} $\subset$ GM^{cu}. Finally, if there exists no GM^{r} such that GM^{r} $\subseteq$ GM^{cu}, we say that the string is ungrammatical (with respect to **R**).

Table 2 shows the GM^{cu} of a locally ambiguous but globally unambiguous string. Node x_4 cannot be directly governed by both x_3 and x_5, hence the string is locally ambiguous. Only the former choice results in a totally governed string (Figure 1 and Table 1).

Node/Relation	*R*$_1$	*R*$_2$	*R*$_3$	*R*$_4$
x_1				
x_2	x_3			
x_3		x_1		
x_4			x_3	x_5
x_5			x_4	

Table 2: Locally ambiguous but globally unambiguous GM^{cu}

Table 3 shows the GM^{cu} of a both locally and globally ambiguous string. Figure 1 shows one and Figure 2 shows another governed string corresponding to this GM^{cu}. If a string is ambiguous, at least one row has multiple entries in the GM^{cu}.

Node/Relation	*R*$_1$	*R*$_2$	*R*$_3$	*R*$_4$
x_1				
x_2	x_3			x_1
x_3		x_1		
x_4			x_3	
x_5			x_4	

Table 3: Locally and globally ambiguous GM^{cu}

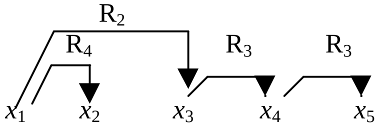

Figure 2: Another governed string

1.3 Deterministic parsing

An GM^{r} carries all necessary information about the structure of a governed string. If the process of uncovering the structure of a string is called **parsing**, a parsing process equals to the finding of the GM^{r} for a given string (or all resolved maps if the string is globally ambiguous), and the found GM^{r} represents the **parse tree** of the string.

The nodes and relations in a GM generate an abstract **search space** for governed strings. Therefore, parsing can be viewed as a **search** for the GM^{r} in the space genereted by the string of nodes and the set of available relations. The process begins with an empty map and makes progresssively more and more direct governors known. The process should end with a GM^u such that $GM^r \subseteq GM^u$. If $GM^r \subset GM^u$ there remains a residual problem of finding $GM^{u'}$, $GM^{u'} \subset GM^u$, such that $GM^{u'} = GM^r$.

Let us assume that for each globally ambiguous string there is single right parse tree, called the **preferred** tree. We call a parsing process **deterministic**, if it begins with an empty map and marks direct governors in the map in such an order that when the process ends $GM^u = GM^r$, where GM^u is the explored map and GM^r represents the parse tree or the preferred parse tree if the string is globally ambiguous.

Theorem 1: Unambiguous strings can be parsed deterministically.

A proof is trivial. Any algorithm which finds all possible direct governors of the nodes by iterating through all the relations and all the nodes creates the GM^{cu} by definition. And with unambiguous strings, $GM^r = GM^{cu}$.

The following OS algorithm (for Open Search), among others, parses unambiguous strings deterministically.Let n_R denote the number of available relations and n_N stand for the number of nodes in an input string.

OS algorithm:

1. Assign the available relations as columns in a GM in random order.
2. Assign the nodes of an input string as rows in the GM in their precedence order.
3. Mark each cell in the GM empty and each row open.
4. For each column k (k=1, ..., n_R) test each x_i (i=1, ..., n_N) and each open x_j (j=i-1, i-2, ..., 1, i+1, i+2, ..., n_N) for x_i R x_j, where R is the relation assigned in the column k. Mark each found direct governor x_i in the GM[j,k] and mark the row j closed.

Let us call the number of open nodes (plus 1) between a direct governor and the governed node at the moment of a test the **distance** of the relation test.

Distance Hypothesis: It is possible to order linguistic dependency relations as columns in a GM in such a way that the maximum distance remains within a fixed boundary when the OS algorithm parses natural language sentences. (We return to this hypothesis later on in this paper.)

Theorem 2: If the distance hypothesis holds, unambiguous strings can be parsed in linear time.

Let us assume that the distance hypothesis holds and let d stand for the maximum distance. The iteration statement in the OS algorithm is then limited as follows:

```
…
(j=i-1, j-2, ..., i-d, i+1, i+2, ..., i+d);
…
```

Let C denote the most expensive relation test. The OS algorithm consumes in the worst case at most $C * n_R * n_N * 2 * d = O(n_N)$.

Next we show that even ambiguous natural language sentences can be parsed deterministically in linear time if a certain additional condition holds.

Best-First Conjecture: It is possible to order the linguistic relations as columns in a GM in such a way that (without violating the Distance Hypothesis) the OS algorithm produces for natural language sentences the right or the preferred GM^r most of the time.

Due to its heuristic flavor, we call the thus modified OS algorithm the BF algorithm.

BF algorithm:

1. Assign the available linguistic relations as columns in a GM in such an order that both the Best-First Conjecture and the Distance Hypothesis hold.
2. (steps 2.-4. are as in the OS-algorithm)

The enforcement of the Best-First Conjecture brings a heuristic component in the algorithm, and the algorithm does not explore the search space fully anymore. Once the algorithm chooses a local governor over the alternative ones for a word, the alternative local governors will be rejected forever. Therefore, there is no guarantee that the right parse trees will be always produced, hence the phrase “most of the time”.

Claim: The BF algorithm parses natural language sentences deterministically in linear time so that the right or the preferred parse trees are produced most of the time.

This claim is an unprecise empirical statement that can be supported only by empirical means. That will be done next.

2 The Practical Parser

From now on we assume that strings of nodes are natural language sentences and discuss a fully implemented parser (DCParser) that parses Finnish sentences. The DCParser differs from the simple theoretical model described above, but, as will be shown below, the differences do not alter the theory.

2.1 Contexts

The formal part introduced binary relations as context-free ordered pairs (1). Dependency relations in the implemented parser use contexts. Formally, they could be expressed as context-sensitive ordered pairs as in (2), but the DCParser uses different rule syntax as discussed in 2.5.

(2) $R_i = \{ < [cx_l]\ \mathbf{x}\ [cx_r], [cy_l]\ \mathbf{y}\ [cy_r] > |$
$\mathbf{x}$, $\mathbf{y}$ are morphosyntactic representations of the direct governor and the governed word form,
cx_l, cx_r, cy_l, cy_r are morpho-syntactic representations of the left and the right contexts of $\mathbf{x}$ and $\mathbf{y}$, respectively,
and $\mathbf{x}\ R_i\ \mathbf{y}\ \}$.

The use of contexts in relations adds another heuristic component to the BF-algorithm, and one dependency relation may require quite a few but fixed number of such context sensitive definitions. Contexts do not alter, however, the linear time behavior of Theorem 2. They only increase the value of the constant C (the computational cost of the most expensive relation test).

2.2 Decomposition

The theoretical model assumes that sentences are parsed in one pass. The DCParser divides sentences into segments, using conjunctions and delimiters as separators. The BF-algorithm is applied to each segment separately, and the final phase unites the structures built in those segments applying the algorithm again. Decomposition greatly strengthens the Distance Hypothesis, but it does not alter the linearity proof, since the sum of linear elements is linear:

(3) $O(n_i) + O(n_j) + ... + O(n_k) = O(n_N)$,
$n_i, n_j, \ldots, n_k \leq n_N$

where n_N is the number of the words in a sentence.

2.3 Homographic disambiguation

The theoretical model did not discuss ambiguous nodes. In practice a word form can have several alternative morphotactic interpretations. The DCParser has a separate morphological analysis phase which produces all possible morphotactic interpretations for the word forms of input sentences. A separate preprosessing phase explicitly disambiguates most of the lexical and homographic ambiguities of Finnish word forms using context sensitive rules designed for the purpose (Nykänen, 1986). The remaining ambiguities are resolved implicitly by the DCParser as follows. When an interpretation of an ambiguous word form qualifies as a governed node the alternative interpretations will be rejected. This strategy implements yet another heuristic component for the parser, but the strategy does not alter the linearity argument presented earlier.

2.4 The dependency relations

The parser uses 32 different binary dependency relations for Finnish. The coordinating relations are discussed in 2.5. The most important other relations are listed in Table 4. The typical syntactic categories for the regents and for the dependants are also shown. Space does not allow a discussion of the individual relations. They are visualized in examples below. By stipulation, the finite verb of the main clause is the head of a grammatical sentence.

Relation name	Dependant	Regent
IntensAttr	Adverb	Adverb/Adj.
ModAttr	Adverb	Noun/Adj.
AdjAttr	Adjective	Noun
GenAttr	Noun	Noun
QuantAttr	Num./Adv./Pron.	Noun
NomAttr	Noun	Noun
MaterAttr	Noun	Noun
InfAttr	Verb	Noun
RelAttr	Verb	Noun
ClauseAttr	Verb	Noun
PostpComp	Noun	Postposition
PrepComp	Noun	Preposition
NegComp	Verb	NegVerb
AuxComp	Verb	Copula
Subject	Noun	Verb
Object	Noun	Verb
Adverbial	Noun/Adverb	Verb
Complement	Noun/Adjective	Copula
Connector	Delimiter	Verb/Noun
Separator	Delimiter	Verb
Head	Verb	none

Table 4: Common relations

2.5 Coordinations

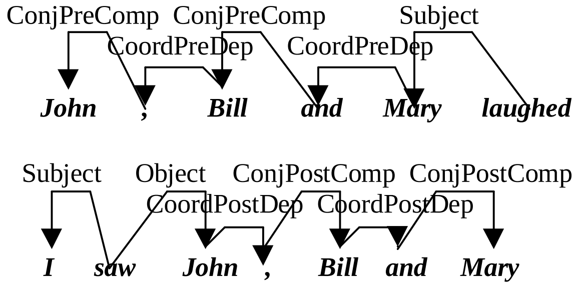

Figure 3: Coordinations

Coordinations are one of the main sources of syntactic ambiguity in natural language sentences. For us they cause also a notational problem, since coordinations do not seem to be prima facie binary relations. The DCParser treats a coordination as two coexisting binary relations. One word governs the coordinator which governs the other word. By stipulation, that word among coordinated words which is closest to the regent becomes the head of the coordination. For example, the coordinated subject in the sentence *John, Bill and Mary laughed* is ascending, while the coordinated object in the sentence *I saw John, Bill and Mary* is descending as Figure 3 illustrates.

2.6 Subordinate clauses

The DCParser treats finite subordinate clauses so that the subordinating conjunction serves as a linking word between the heads of the main and the subordinate clauses. The conjunction is in the relation in question, and the head of the subordinate clause is in the ConjPostComp-relation with the conjunction. Below there is a Finnish example sentence from the corpus, its rough word-for-word translation and the parse tree produced by the DCParser (4). This sentence exemplifies both subordinate clauses and coordinations. In this output mode the DCParser displays word forms as triplets: *surface form*, **Relation**, base form. Hierarchy is indicated using indentation: the regent of a given dependant is the first word below that is indented one step less.

Riittää, kun puolueet ja niiden järjestöt
[It is enough], [when] [the parties and their organiz.]
velvoitetaan lainsäädännön avulla julkaisemaan
[are compelled] [using legislation] [to publish]
tarkasti tilinpäätökset, budjettinsa ja
[accurately] [financial statements, their budgets and
lahjoituksensa.
their donations].

(4)

```
 |-,, Connector, _COMMA
 |       |-puolueet, ConjPreComp, puolue
 |     |-ja, CoordPreDep, ja
 |     |-niiden, GenAttr, ne
 |   |-järjestöt, Object, järjestö
 |   | |-lainsäädännön, GenAttr, lainsäädäntö
 |   |-avulla, Adverbial, apu
 |   |         |-lahjoituksensa, ConjPostComp, lah...
 |   |       |-ja, CoordPostDep, ja
 |   |     |-budjettinsa, ConjPostComp, budjetti
 |   |   |-,, CoordPostDep, _COMMA
 |   | |-tilinpäätökset, Object, tilinpäätös
 |   | |-tarkasti, Adverbial, tarkasti
 |   |-julkaisemaan, Adverbial, julkaista
 | |-velvoitetaan, ConjPostComp, velvoittaa
 |-kun, Adverbial, kun
 |-., Separator, _PERIOD
Riittää, Head, Riittää
```

Another sentence from the corpus and its parse tree are as follows:

Kysymys *askarruttaa koko maailmaa* *nyt*,
[The question][puzzles] [the whole world][now,]
kun *Yhdysvaltain republikaanit*
[when] [the republicans of the U.S.]
ovat pitäneet puoluekokouksensa
[have held] [their party congress]
ja *nimenneet* *presidentti Bushin*
[and] [nominated] [president Bush]
ja *varapresidentti Dan Quaylen*
[and] [vice-president Dan Quayle]
taisteluparikseen,
[as their fighting couple,]
kuten neljä vuotta sitten.
[like] [four years ago.]

(5)

```
  |-Kysymys, Subject, Kysymys
  |  |-koko, AdjAttr, koko
  |-maailmaa, Object, maailma
  |  |-,, Connector, _COMMA
  |  |        |-Yhdysvaltain, GenAttr, Yhdysvallat
  |  |     |-republikaanit, Subject, republikaani
  |  |     |  |-puoluekokouksensa, Object, puoluekokous
  |  |     |  |        |-presidentti, NomPreAttr, presidentti
  |  |     |  |        |        |-varapresidentti, NomPreAttr, ..
  |  |     |  |        |     |-Dan, NomPreAttr, Dan
  |  |     |  |        |  |-Quaylen, ConjPostComp, Quayle
  |  |     |  |        |-ja, CoordPostDep, ja
  |  |     |  |     |-Bushin, Object, Bush
  |  |     |  |     |  |-,, Connector, _COMMA
  |  |     |  |     |-taisteluparikseen, Adverbial, taistelu...
  |  |     |  |     |        |-neljä, QuantAttr, neljä
  |  |     |  |     |     |-vuotta, PostpComp, vuosi
  |  |     |  |     |  |-sitten, ConjPostComp, sitten
  |  |     |  |     |-kuten, Apposition, kuten
  |  |     |  |  |-nimenneet, ConjPostComp, nimetä
  |  |     |  |-ja, CoordPostDep, ja
  |  |     |-pitäneet, AuxComp, pitää
  |  |  |-ovat, ConjPostComp, olla
  |  |-kun, ClauseAttr, kun
  |-nyt, Adverbial, nyt
  |-., Separator, _PERIOD
|-askarruttaa, Head, askarruttaa
```

2.7 The grammar

In the DCParser word forms are represented as **objects of morpho-syntactic attributes**. For example, the word form *järjestöt* (organizations) appears as [Form="järjestöt", Lex="järjestö", Cat=Noun, Case=Nom, Number=PL]

For efficiency reasons binary relations are expressed as **active rules**. The testing of a relation, then, corresponds to the activation of the respective rule or a set of alternative rules. For example, a simplified rule for AdjAttr (adjectival attribute of nouns) reads as:

(6)

```
-- Rule for adjectival attributes
Redo AdjAttr
Node Focus [ Cat=Noun, Cs:=Case, Nm:=Number ]
D1 := DepCand Left(1) [ Cat=Adjective,
                        Case=Cs, Number=Nm ]
        -> MakeDep D1 [ Rel:=AdjAttr ]
```

A rule has two main parts: the condition part and the action part. The condition part searches and tests qualifying dependants and possible contextual words. A word qualifies in a test if its attribute object satisfies the description given in the rule. Variables can be used for passing attribute values. (":=" assings a value; "=" tests a value) The action part binds and names dependants and assigns values to attributes. The rule above iteratively (Redo) binds immediately preceeding adjectives as attributes if they agree in the case and number with the head noun.

Rules are classified into **generic** rules (grammar proper) and **lexical** rules. Their expressive power is identical. The former are activated by syntactic categories. (6) visualizes a simple generic rule. Lexical rules are activated by specific lexemes. For example, (7) describes a part of a complex rule for Finnish verb *pitää*.

(7)

```
LexBlock "pitää"
-- pitää tehdä
Once pitaa1
D1 := DepCand Right(4) [ Modal=Iinf ]
        -> MakeDep D1 [ Rel:=Subject ]
           Focus [ SubCat:=InfSubj ]
...
-- pitää jostakin
Once pitaa10
D1 := DepCand Right(4) [ Cat=Noun+Proper+
                              Pronoun,
                         Case=El ]
Not Node From D1 Right(1) [ Modal=Ipartic+
                                  IIpartic ]
        -> MakeDep D1 [ Rel:=Adverbial ]
           Focus [ SubCat:=Intr ]
```

Pitää has several senses and subcategories in Finnish. (7) shows two of them. The first alternative treats the verb as a modal verb as in *Minun pitää mennä saunaan* (I must go to the sauna). (In our linguistic analysis we treat the infinitive *mennä* (to go) as the subject of the modal verb *pitää* and the genitive *minun* (?I) as the subject of the infinitive.) The second alternative handles the idiomatic usage *Minä pidän hänestä* (I like her) where a surface elative adverbial represents a deep semantic object of *pitää*. The rule binds an elative as adverbial, but does not bind it if the elative is followed by a participle as in *Minä pidän hänestä lähtevästä tuoksusta* (?I like the fragrance coming from her).

The grammar (Arnola, 1998) consists of about 950 generic rules and of about 12 500 lexical rules. An algorithm, which implements the Best-First strategy, controls the activation of the rules.

3. Empirical Results

3.1 Benchmark test suite

The parser has been under development for years. It is an integral part of a commercial machine translation system called TranSmart®. A benchmark test suite of correctly parsed sentences (source sentences and their correct parse trees) has been accumulated during this period. Only sentences that have revealed grammatical errors in the parser have been added to the test suite after the errors were corrected. Otherwise the test suite sentence have been randomly selected. The test suite sentences are periodically parsed to guarantee monotonous improvement of the grammar.

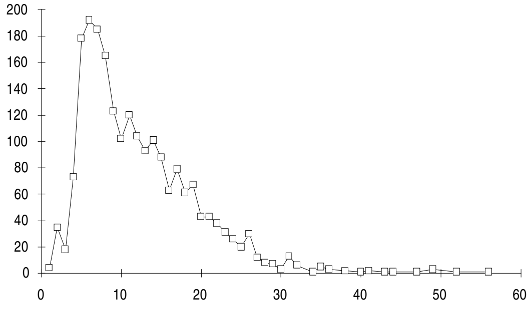

Figure 4: Distribution of the sentence lengths of the benchmark test suite

As of this writing, the benchmark test suite comprises over 3000 sentences. The distribution of the sentence lengths (including delimiters) is shown in Figure 4. The average sentence length is 12.1 words.

3.2 Linearity argument

We used the benchmark test suite sentences to test the linearity claim. Figure 5 shows the distribution of the parsing times in seconds. The processor is an old Intel 486, 66 MHz. A 150 MHz Pentium processor parses about 400 sentence/minute of running text.

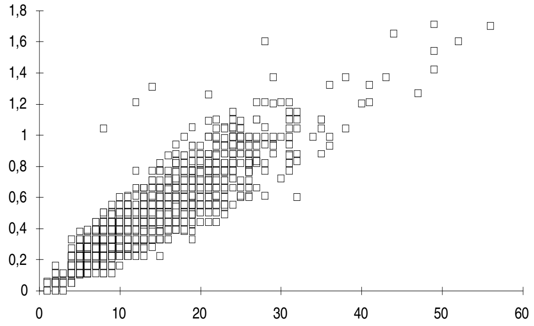

Figure 5: Distribution of the parsing times of the test suite sentences

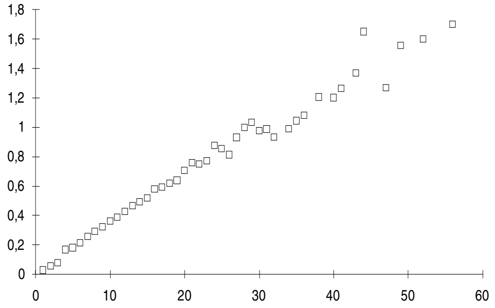

Figure 6: Average parsing times for different sentence lengths

Figure 6 plots the average parsing times for each sentence length. Sentences whose length is between 5 and 20 words form statistically meaningful sets. Their average parsing times form a clear linear function. Longer sentences do not support a contrary view.

3.3 Quality

It remains to discuss the quality of the parser. We use the following strict criterion for the correctness of a parse tree. A sentence is parsed correctly if the sentence is grammatical and the produced dependency structure completely complies with the structure a competent human judge would assign to it. Otherwise the parse tree is judged incorrect. Hence, a single, local structural error in an otherwise correct parse tree disqualifies the structure. If a sentence is globally ambiguous but it is clear for a human reader which structure is meant, the structure is judged correct only if it is in agreement with the human decision. If a human reader cannot make the right choice for an ambiguous sentence without textual context, the structure is deemed correct if it is one of the possible correct structures.

Presently the DQParser is fully developed in the sense that it is in practical use in commercial machine translation systems. However, the tuning of the parser still continues. The parser has been subjected to tens of thousands of genuine unedited sentences from different sources over the years. Each parse tree has been carefully studied and all indicated errors or gaps that could be systematically corrected were corrected in the grammar and in the lexicons. About once a week the benchmark test suite was processed and possible errors found in the test suite were corrected.

Occasionally (about once in a month or two) a fresh piece of text was randomly selected. The total number of sentences in the text and the number of sentences parsed correctly right away were recorded. The incorrectly parsed sentences were classified into three classes: the ones parsed correctly after (only) lexical corrections, the ones parsed correctly after grammatical corrections (and possible lexical corrections), and the ones whose parsing errors could not be corrected in a systematic fashion. These errors exhibit a fundamental drawback of the Best-First strategy. Table 5 shows the data of these test samples. Each column presents both absolute and relative numbers: absolute/percentage%.

Text	No. of sent.	Parsed correctly	Req. lexical correct.	Req. gramm. correct.	Fatally in-correct
1	375	294/78%	50/13%	19/5%	12/3%
2	196	148/76%	34/17%	8/4%	8/4%
3	196	162/83%	18/9%	10/5%	6/3%
4	181	148/82%	19/10%	6/3%	8/4%
5	253	207/82%	26/10%	11/4%	9/4%
6	380	331/87%	38/10%	4/1%	7/2%
7	216	174/81%	27/13%	8/4%	7/3%
8	297	249/84%	33/11%	8/3%	7/2%
9	387	334/86%	34/9%	11/3%	8/2%
10	196	166/85%	16/8%	8/4%	6/3%
11	267	224/84%	25/9%	8/3%	10/4%
12	118	97/82%	16/14%	2/2%	3/3%
13	2680	2271/85%	252/9%	83/3%	74/3%
14	391	337/86%	36/9%	11/3%	7/2%

Table 5: Parsing quality of the test samples

Figure 7 shows the percentage numbers of each column in a graphic form. Lines are fitted to the data to indicate possible tendencies of the series.

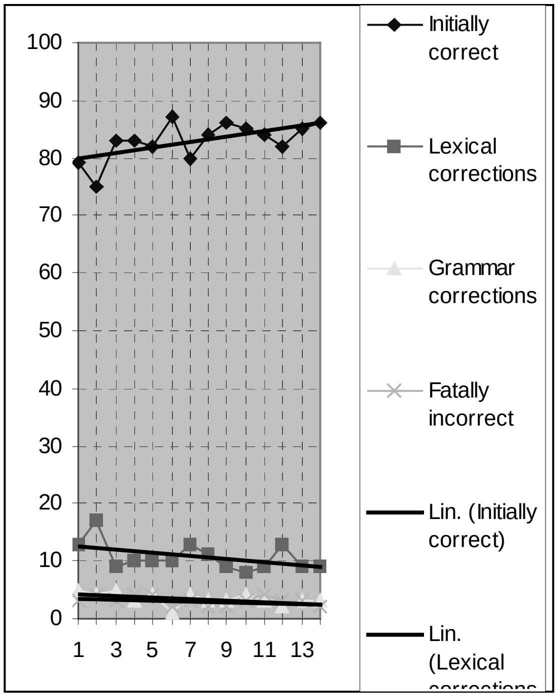

Figure 7: Parsing quality of the 14 last samples

Table 5 and Figure 7 show that the parser seems to embody a stable 2-4% error ratio due to fundamental problems in the Best-First strategy. Approximately the same number of sentences (2-5%) have revealed grammatical deficiencies in the parser. This figure may have a slow, although not clear declining trend. 9-17% of the sentences have revealed lexical deficiencies, and this figure seems to have a slow declining trend. 76-87% of the sentences were parsed correctly right away, and this figure seems to show a clear if slow upward trend. (The test samples cover almost two years of rather intense tuning.)

Conclusion

In this paper we have argued that it is possible to parse binary dependency structures of natural language sentences deterministically and in linear time, and to keep parsing quality within acceptable limits, if syntactic heuristics is applied appropriately. A possibility for linear parsing has been proved theoretically and demonstrated empirically. The quality issue was discussed using empirical data. Determinism was accomplished with a Best-First search algorithm which implements syntactic heuristics in three ways: 1) in a permanent ordering of the testing of dependency relations, 2) in the implicit disambiguation of homographic word form interpretations, and 3) in the contexts of dependency relation rules.

Linear behavior is strongly supported by the empirical data. It is difficult to be precise about the quality issue. Empirical data shows that the upper limit of the quality of this deterministic strategy is 96-98%. The inherent error rate is due to the use of heuristics. Nondeterministic parsers do not have such theoretical barriers. But this inherent error ratio should be contrasted with the fact that a deterministic parser produces the right parse tree, while a nondeterministic parser produces usually only a forest of candidate parse trees accurately.

At the moment of this writing this deterministic parser seems to have reached about 85% correctness rate (the average of the last five samples). Current errors are mainly lexical errors or gaps (about 9%) which usually can be easily corrected but the corrections improve the quality only slightly. Some 3% of the current errors are errors and gaps in the grammar. One should be cautious, however, of giving any precise numbers for parsing quality, since our exprerience shows that quality numbers vary markedly from one text to another.

An interactive demonstration of the parser is available to the public for testing purposes at http://www.kielikone.fi/dcparser-fi-demo, and the machine translation system (from Finnish into English) at http://www.kielikone.fi/fealcee.

Acknowledgements

My thanks go to the whole personnel of Kielikone Ltd. and, in particular, to Kaarina Hyvönen, Jukka-Pekka Juntunen, the late Tim Linnanvirta, and Asko Nykänen, who have contributed to the paper. I also want to thank the anonymous referees of this article. The Sitra Foundation and The Technology Development Centre have financially supported the work reported in this article.

References

Arnola, H. (1998) The functional dependency structure of Finnish. (manuscript)

Hays, D. (1964) *Dependency theory: a formalism and some observations.* Language, 40, pp. 511-525.

Hellwig, P. (1986) *Dependency unification grammar.* Proc. COLING-86, Bonn.

Hudson, R. (1976) *Arguments for a Non-transformational Gramma*r. The University of Chicago Press.

Jäppinen, H., Lehtola, A., and Valkonen, K. (1986) *Functional structures for parsing dependency constraints*. Proc. COLING-86, Bonn, pp. 461-463.

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

Nelimarkka, E., Jäppinen, H., and Lehtola, A. (1984) *Two-way automata and dependency grammar: a parsing method for inflectional free word-order languages*. Proc. COLING-84 and 22th ACL Meeting, Stanford, pp. 389-392.

Nykänen, A. (1996). *Design and Implementation of an Environment for Parsing Finnish.* M.Sc. (Eng.) Thesis, Helsinki University of Technology, Department of Computer Science (in Finnish)

Robinson, J. (1970) *Dependency structure and transformational rules*. Language, 2/46.

Schubert, K. (1986) *Linguistic and extra-linguistic knowledge.* Computers and Translation, 1, pp. 125-152.

Starosta, S. (1986) *The Case for Lexicase*. Pinter Publisher.

Valkonen, K., Jäppinen, H., and Lehtola, A. (1987) *Blackboard-based dependency parsing*. Proc. IJCAI-87,Milan, pp. 700-702.