A Principle-based Parser for Foreign Language Training in German and Arabic[*]

Joe Garman[*] Jeffery Martin[*] Paola Merlo[†] Amy Weinberg[*]

[*] Dept. of Linguistics and Institute for Advanced Computer Studies
University of Maryland at College Park, U.S.A.
email: {garman|jeffmar|weinberg}@umiacs.umd.edu

[†] Dept. of Linguistics and Dept. of Psychology
University of Geneva, Switzerland
email: merlo@divsun.unige.ch

Abstract

In this paper we discuss the design and implementation of a parser for German and Arabic, which is currently being used in a tutoring system for foreign language training. Computer-aided language tutoring is a good application for testing the robustness and flexibility of a parsing system, since the input is usually ungrammatical in some way. Efficiency is also a concern, as tutoring applications typically run on personal computers, with the parser sharing memory with other components of the system. Our system is principle-based, which ensures a compact representation, and improves portability, needed in order to extend the initial design from German to Arabic and (eventually) Spanish. Currently, the parser diagnoses agreement errors, case errors, selection errors, and some word order errors. The parser can handle simple and complex declaratives and questions, topicalisations, verb movement, relative clauses — broad enough coverage to be useful in the design of real exercises and dialogues.

1 Introduction

This paper describes the design and implementation of a parser for German and Arabic, which is currently being used in a tutoring system for foreign language training. Computer-aided language tutoring is a good application for testing the robustness and flexibility of a parsing system, since the input is usually ungrammatical in some way. The system is portable in that we extend the initial design for German to handle Arabic and (eventually) Spanish. Efficiency is also a concern, since tutoring applications typically run on personal computers, with the parser sharing memory with other components of the system. Our system is principle-based, which ensures a compact representation.

Robustness is required because the system must be able to parse incorrect input. Since the user is not a native speaker of the language, the system must handle spelling errors such as (1a) as well as grammar errors, as in (1b).

(1) (a)

Der	*Fluß*	*flest*	*auf*	*Westen*
The	river	flows	to	west

(b)

Der	*Fluß*	*fließen*	*auf*	*des*	*Westen*
The	river	flow.pl	to	the	west.gen

'The river flows to the west'

[*]The names of the authors are in alphabetical order. This paper is the result of cooperative work in all its stages. However, as far as legal requirements are concerned, Paola Merlo takes responsibility for sections 4 and 5.

(2) (a)

at-tariiqu	*taqa9u*
road.sg.def	locate.3.f.sg

(b)

aT-Tariiqu	*taqa9aayna*
road.sg.def	locate.3.f.pl

'The road is there.'

The system must be flexible because it must be able to handle different levels of ill-formedness. For example, it must be able to detect the difference between (1a) and (1b). In the former the verb *flie;ßt* is incorrectly typed, while in the second sentence the verb is incorrectly inflected and the wrong case is selected for the prepositional phrase, which should be *auf Westen*. The sentences in (2) illustrate a similar pair from Arabic. In (2a) the word for *road* should have an emphatic consonant (signified by a capital letter). In (2b), there is an agreement error (the subject is third feminine singular, while the verb is third feminine plural). The former type of mistake must be corrected for the parser to succeed. This correction is done by an interaction with the user. The latter type of error is bypassed by a special mechanism of defaults that enable the parser to analyse some kinds of incorrect input.

Portability in a language processor means that the design can be reused for different languages without fundamental changes. The dictionaries and some of the grammatical information will be different, but it is desirable to design the Natural Language Processing (NLP) component in the most general way.[1] Arabic and German are good bounding cases to test the portability of the design. The languages are similar in having rich inflectional morphology and in having various forms of grammatical agreement. They are different enough to constitute a fair test of the portability of the system. For instance, they have different word orders: German is basically verb-final, with movement of the verb in second position in main clauses, while Arabic has SVO word order, with VSO occurring frequently in surface order. Grammatical subjects are usually dropped in Arabic and expressed by agreement with the verb, while in German null-subject sentences are not grammatical. Small clauses are extremely common in Arabic, and usually copular verbs are not expressed, while German sentences present a clear distinction between predicates (verbs) and arguments (nouns). Thus, for a parser to be applicable to both languages, it must make use of very abstract properties of the grammar.

Finally, the system must be efficient. Ideally, it must be able to parse a sentence, detect the errors and produce an output in the same amount of time it would take a human tutor to perform the same task, otherwise the time lag would decrease the student's attention and motivation. Computational efficiency must not be bought at the expense of space compactness though, because the system must fit on a PC with many other programs running at the same time.

The principle-based approach to parsing allowed us to meet all of these requirements. Previous formalisms, for example, the EST version of generative grammar (Chomsky 1965, 1973, 1977 among others) assumed that every construction of a language was syntactically represented by a grammatical rule. Thus big, monolithic grammars needed to be stored and consulted for any parser with reasonably wide linguistic coverage. Recent developments in grammatical theories, in many different frameworks, have succeeded in identifying the unifying principles of many, apparently unrelated, linguistic phenomena.

The Government-Binding (GB) framework (Chomsky 1981) provides construction-independent principles that are grouped into interacting modules. The modules are parameterised, so that by modifying them to a small degree, one can generate the patterns associated with a variety of languages. The modularity of the grammar makes it easy to relax certain constraints and thereby obtain a parse for various types of ungrammatical input. Because language-dependent information is separate from language-independent information, the same parsing design is valid across languages. Finally, the factorization into modules makes the grammar compact, and consequently storage needs are minimised.

Our implementation provides experimental answers to the issues of parsing ill-formed input, generalisation across languages, and compactness of representation. It addresses the problems of modularisation, how much compilation across principles is necessary, and how various principles can be efficiently interleaved. The following sections discuss the actual implementation, illustrate the design criteria, and finally

[1]In the best case the design is explicitly parameterised so that it specifies a family of parsers.

evaluate the performance of the parser. The part of this research which addresses issues related to foreign language training and tutoring systems is mentioned only in discussing the constraints on parsing design imposed by the application. Section 2 provides an overview of the NLP system, while sections 3, 4, and 5 illustrate the routines for the recovery of phrase structure, feature annotation, and error detection, respectively. Evaluation and illustration of coverage are presented in section 6, followed by concluding remarks in section 7.

2 Overview

The Input/Output The input to the parser is an unannotated German or Arabic sentence. Currently the Arabic is in phonetic transcription, but an interface allowing the use of Arabic script is being developed. The output consists of a parse tree which encodes the hierarchical and linear relations between the elements of the sentence (which is passed to a semantic analyser), and a (possibly empty) list of errors in the sentence, which is passed to the tutor. The following examples show sentences with various errors which illustrate the input/output of the parser (omitting the parse trees).

(3)

```
==> Das frauen gestern hat in des berges
geblieben.
Errors: (1) The article "das" does not
agree with "frauen". (2) Word Order: The
verb "hat" should be in second position.
(3) There is a case selection error between
"geblieben" and "in des berges". The verb
"geblieben" is a state verb and takes dative
case. (4) Wrong auxiliary: "hat". The verb
"geblieben" takes sein. (5) The subject "das
frauen" does not agree with the verb "hat" in
person or number.
```

(4)

```
   ==> ayna aqa9u mintaqati d-dibdibbatu ?
Errors: (1) Spelling error: emphatic/non-em-
phatic substitution in "mintaqati" (2) Case
error between "minTaqati" and "d-dibdibbatu"
(3) Verb/Subject agreement error between
"aqa9u" and "minTaqati d-dibdibbatu"
```

These examples show a simple Prolog interface; the tutor interface on the PC is written in an authoring language which is not discussed here. The parse trees are not displayed to the student.

The Components The main components of the parser are a morphological analyser, a syntactic parser, and an error handling facility. The general overview of the system is shown in Figure 1. The parser receives the input sentence and it sends it to an interactive preprocessor which expands possible contractions, such as *zum* into *zu dem*, and checks if the input is correctly typed.[2]

Parsing and morphological analysis perform a single pass on an input sentence. The parser calls the morphological analyser to place words on the parser's buffer stack. Each call to the morphological analyser returns a set containing all the analyses of the next input word, as it may not be possible to determine which among them is correct at the point where the morphological analyser is called. If only one analysis is selected, it may cause the parser to fail in a later state, and require backtracking. The morphological analysers for German and Arabic are somewhat different owing to the differing morphological features of the two languages.

When the morphemes are correctly identified, the word is passed to the syntactic processor. Before entering this stage, each token is projected to its categorial node; for instance, the verb form *fließt* is projected to V. The shift-reduce parsing algorithm operates on a small set of context-free rules, which store only a very limited amount of hierarchical information. Other kinds of information to build the right parse tree and annotate it correctly are stored in a set of constraints that must be satisfied before a rule can apply. We discuss the rules and constraints in the next two sections.

[2]This is the level at which misspellings are corrected. For more details on the preprocessor, routines on lexical search and morphological analyser, see Azadegan *et al.* (forthcoming).

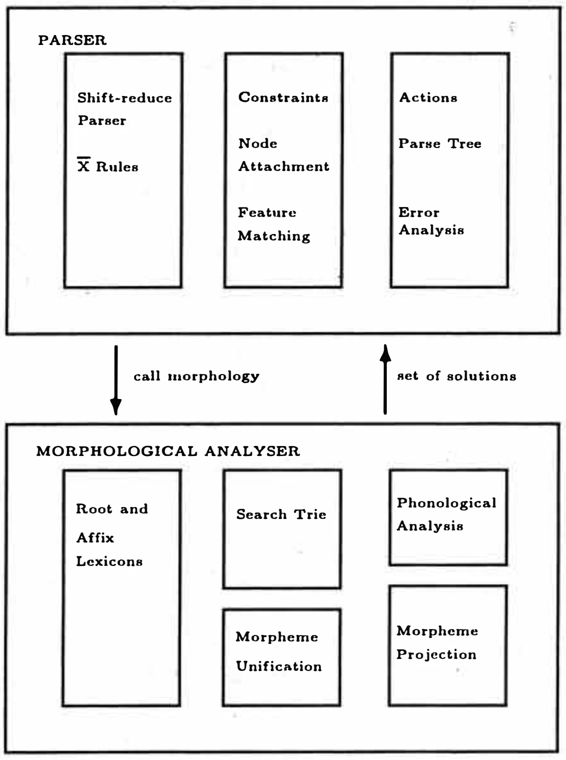

Figure 1: Main Blocks in the Natural Language System

3 The Recovery of Phrase Structure

The parser builds structure by using an augmented shift-reduce method (Aho and Ullman 1972) modified to be more suitable for natural language parsing. Two main modifications are used.

First, the $\overline{\text{X}}$ system of our grammar framework is minimal. Most of the work in parsing consists in constraint checking rather than manipulating $\overline{\text{X}}$ rules. Thus, these rules constitute a context-free backbone anchoring a set of grammar constraints, similar to implementations based on unification grammars, such as PATR-II (Shieber 1986). To separate constraints from phrase structure, we use rules of the following form:

(5) $\text{X} \rightarrow \text{Y Z} \Leftrightarrow f(X, Y, Z)$

The lefthand side is an $\overline{\text{X}}$ production, while the predicate $f(X, Y, Z)$ on the righthand side represents a conjunction of grammar constraints on the features of the nonterminals which must be satisfied to license the production. The form of these constraints and the way they are applied is discussed in the section on constraints.

Second, it is necessary to express movement rules, which relate overt categories to empty categories (also called gaps) in a different position. Our design separates movement rules from $\overline{\text{X}}$ rules. For this purpose a *move-x* operation is introduced in the parse cycle. The use of gaps achieves a greater degree of generalisations in the treatment of superficially different constructions, for instance relative clauses and questions. In order to maintain a limited number of $\overline{\text{X}}$ rules, however, the computation of gaps is performed on-line, by specific routines, rather than precompiled into the rules. These two features of the parser are presented in more detail below.

3.1 The $\overline{\text{X}}$ Rules

$\overline{\text{X}}$ theory is intended to capture universal properties of phrase structure by means of a very small set of rules, which define the phrasal projections and their parts, distinguishing heads from non heads, the latter including complements, specifiers, modifiers, and adjuncts. The $\overline{\text{X}}$ rules refer only to the bar level of the phrase; they do not refer to other grammatical features (Chomsky 1970, Jackendoff 1977, among others). $\overline{\text{X}}$ rules are maximally binary in our system (Kayne 1984). Although they refer to phrases of a specific level, they do not refer to the specific category or features of the phrases. The complete $\overline{\text{X}}$ system employed by the German parser is given by the set of Prolog clauses in (6). The rules for Arabic are given in (7). It is evident from (6) and (7) that very similar sets of rules can handle both languages.

(6) GERMAN:

```
x(2,C)==>[x(2,C1),x(1,C),prespecifier].
x(1,C)==>[x(0,C),x(2,C1),postcomplement].
x(1,C)==>[x(2,C1),x(0,C),precomplement].
x(1,C)==>[x(2,C1),x(1,C),premodifier].
x(1,C)==>[x(1,C),x(2,C1),postmodifier].
x(2,C)==>[x(2,C1),x(2,C),preadjunct].
x(2,C)==>[x(2,C),x(2,C1),postadjunct].
x(1,C)==>[x(0,C),unary1].
x(2,C)==>[x(1,C),unary2].
```

(7) ARABIC:

```
x(2,C)==>[x(2,C1),x(1,C),specifier].
x(1,C)==>[x(0,C),x(2,C1),complement].
x(1,C)==>[x(1,C1),x(1,C),premodifier].
x(1,C)==>[x,(1,C),x(1,C1),postmodifier].
x(2,C)==>[x(2,C1),x(2,C),preadjunct].
x(2,C)==>[x(2,C),x(2,C1),postadjunct].
x(1,C)==>{x(0,C),unary1].
x(2,C)==>[x(1,C),unary2].
```

A term `x(Level,Category)` represents a phrasal projection, where `Level` is the $\overline{\text{X}}$ level taking the values {0,1,2}, and `Category` takes as its value an atomic category symbol from the set {N,V,A,P,Adv,Det,Infl,Comp,Conj}. For example, `x(N,2)` is an NP, and `x(V,1)` is the first projection of V. In $\overline{\text{X}}$ theory, the nonterminal domain is restricted to projections of the atomic category symbols. This restriction holds in (6) and in (7) since the category of a projection is not instantiated in any of the clauses; all categorial information in the system comes from lexical entries themselves. Each righthand side contains a head, which shares the category variable with the lefthand side; a label, which is used to index constraints and tree-building predicates; and possibly a satellite, which is a specifier, complement, adjunct, or modifier.

This design allows no compilation of bar level and category type, at this stage. Berwick (1991) suggests that this organisation slows down the parser, as found experimentally by Dorr (1987). However, we differ from Dorr in two main respects: the rule set we use is much smaller, and the flow of control is serial. Assuming that the average number of conflicts in a compiled grammar is a good indicator of the amount of non-determinism that a shift-reduce parser has to face, Merlo (1992) reports comparisons of LR tables derived from grammars which differ only in the instantiation of the $\overline{\text{X}}$ rules. Rules like those in (6) and (7) are compared to rules of identical format, where the head of the rule is instantiated, for instance, in our notation, `x(2,N) ==> [x(2,C1),x(1,N)]`. Merlo finds that instantiation of heads expands the grammar, of course, but does not reduce the number of conflicts.[3] This leads us to think that non-determinism is not affected by adding category information to the rules, unless filtering constraints, such as co-coccurrence restrictions, are also added. Co-occurrence restrictions, however, and their linear order, are language-specific. Therefore, we have chosen to check category information on-line, at the same time as categorial co-occurrence.

From the point of view of grammar engineering, this choice has the advantage of keeping the basic types of information distinct: topological (configurations), lexical, and long distance relations, thus reflecting the information structure of the theory more directly.

[3]In fact the number of conflicts per state increases from an average of 4.2 for $\overline{\text{X}}$ grammars, to 10.3 for instantiated grammars.

3.2 The Shift-Reduce Algorithm

The parser uses three primary data structures: a main stack, a buffer stack, and a hold store which contains copies of moved phrases. In addition, it takes a token stream (the input) and outputs an annotated tree. The recursive parse procedure thus has five arguments:

(8) parse(Stack, Buffer, Hold, Tree, Stream)

A parse configuration consists of a 4-tuple $(Stack, Buffer, Hold, Stream)$. Since the tree is used only for output, it is not included in the configuration. Various operations including reductions, movement rules, shift, and accept, are used to go from one configuration to the next. It is common in shift-reduce parsing to enter a configuration from which several operations are possible. Our algorithm chooses one operation among those available by giving priority to some operations over others. To choose which operation to apply in a given configuration, the algorithm uses Prolog's control strategy (depth-first search) to select the first operation that matches the configuration. Each operation is implemented by a Prolog clause, and the clauses are ordered as in (9).

(9)
1. morphological analysis
2. accept
3. attention shift
4. move-x
5. reduce X = Y Z (binary reductions)
6. reduce X = Y (unary reductions)
7. pop
8. shift
9. fail

Nondeterminism caused by lexical and structural ambiguity is handled in the current system by backtracking. Despite the worst case time complexity, we have found that this approach gives good performance in practice. We are currently experimenting with the use of a graph-structured stack (Tomita 1985), which allows all parser operations to be computed in parallel.

The parser accepts sentence fragments (NP's, VP's, PP's, and so forth) as well as full sentences.[4] The movement clause determines whether a movement rule should apply in a configuration. This operation is discussed in more detail in the next subsection. Binary reductions are performed by the clause in (10). In this clause, the parser attempts to match the top two elements on the stack with the righthand side of a binary rule of the form:

(10) `x(L,C)==>[x(L1,C1),x(L2,C2),Rule]`

The name of a rule, here indicated by the parameter `Rule`, is used to index the constraints which must apply to the feature sets of the elements on the stack if the rule is to succeed. If a rule matches but the constraints for that rule fail, then an attempt is made to match other rules. If the constraint succeeds, tree-building actions are performed and the parse continues with the reduced phrase on top of the stack. An unconditional shift simply moves the top element from the buffer onto the main stack.

3.3 The Move-x Component

Two basic types of movement rules are implemented. The first involves movement of a maximal projection. The most common instance of this type is movement of a question phrase to the sentence-initial position, as in the German example (11a), and in the Arabic example (11b), where the question word (*ayna*) is also moved to the front of the sentence. This type of movement can also generate topicalisations, clefts, and some relative clauses in both Arabic and German.

(11)

(a)

In	*welcher*	*Richtung*	*fließt*	*die*	*Donau?*
In	what	direction	flows	the	Danube

'In which direction does the Danube flow?'

(b)

Ayna	*taqa9u*	*maHaTTatu*	*sh-shurTati*	*nisbatan*	*li*	*T-Tariqi*
where	located	station	police	relation	to	road.def

'Where is the police station in relation to the road?'

[4]This is needed for exercises where the appropriate response is a sentence fragment.

The second type of movement involves movement of a level 0 phrase (a head), which occurs in the yes/no question in (12), where the verb *darf* (*must/may*) has moved.

(12)

Darf	*ich*	*hier*	*bleiben?*
may	I	here	stay

'May I stay here?'

Following Thiersch (1978), the underlying position of the verb in German is the sentence-final position. This is the position occupied by the verb in a subordinate clause, as in (13).

(13)

Ich	*weiss*	*nicht*	*ob*	*ich*	*hier*	*bleiben*	*darf.*
I	know	not	if	I	here	stay	might

'I don't know if I may stay here.'

To form a yes-no question, the verb is first moved to second position, and from there to the clause-initial position. This type of movement is also used frequently in Arabic, where along with the standard Subject Verb Object word order, the Verb Subject Object order is obtained by moving the verb to the front of the sentence. An example is given in (14).

(14)

Taqa9u	*maHaTTaatu*	*shurTati*	*d-dibdibbati*	*9alaa*	*Tariiqin*
located	station	police	def.Dibdibba	on	road

'The Al-Dibdibba police station is located on a road.'

The parser uses the two steps in (15) to relate the surface position of a phrase to its underlying position.

(15)
- Stack-to-Hold Rules: put a trace of the antecedent onto the hold store
- Hold-to-Stack Rules: move a trace from the hold store onto the stack.

Each type of movement rule uses a data structure called the Hold Store (Wanner and Maratsos 1978), which is used for temporary storage of the moved element. The Hold Store contains two cells, one for level 2 phrases (X2HOLD) and one for level 0 phrases (X0HOLD). The use of a Hold store has been criticised from a psycholinguistic and linguistic point of view (Berwick and Weinberg 1984, among others). However, recent work on the connectivity of natural languages (Stabler 1993) suggests that allocating a (small) finite number of *memory registers* to each type of linguistic entity that undergoes long-distance relations captures a wide-spread generalisation, both in syntax (questions, causatives) and morphology (applicatives, datives).

For the Stack-to-Hold operation, we must identify when a phrase is not in its underlying position; *i.e.* we must locate the antecedent. Since the type of positions to which a phrase may move is limited and predictable, the parser consults a table of move-x configurations to determine that a phrase is not in its underlying position.

For instance, the rule applies in a configuration where the stack contains the single *wh*-phrase *in welcher Richtung* ('in what direction') and where the buffer contains the verb *fließt* ('flows'). In such a configuration, a trace of *in welcher Richtung* is placed on the X2 hold store. The trace is identical to the antecedent in all features except the spelling, since the trace does not appear in the surface form.

For the Hold-to-Stack operation, we must find the underlying position of the trace on the hold stack. Again the parser consults a table of configurations which determines whether the trace should be moved from the buffer to the stack. For instance, this rule would apply in a configuration where the verb *fließt* is on top of the stack, and the trace of the phrase *in welcher Richtung* is in the X2 hold store. There is a condition on the rule which requires that the moved phrase be a possible complement of the verb. In this instance, the condition on the rule is satisfied, since *in welcher Richtung* is a possible complement of the verb *fließen.* In this configuration, the trace is moved from the hold store to the second position on the stack. It is moved to second position because complements in German occur on the left of the verb.

We should underline once again that we have adopted a direct implementation of the principles of the grammar. Other approaches (Dorr 1993, Fong 1991) precompile the possible positions where an empty category can occur. Using slash rules (Gazdar *et al.* 1985), in the $\overline{\text{X}}$ schema would amount to the same compilation. Clearly, our approach can lead to frequent postulation of empty categories in many positions that are not licensed by the grammar.[5] Thus, this method of recovering phrase structure information must be coupled with filtering constraints that weed out incorrect derivations as soon as possible. We illustrate the solution adopted for this problem of interleaving the principles in the next section.

4 Constraints

As a result of using an $\overline{X}$ backbone to parse, most of the feature percolation and feature annotation that could be encoded in the nonterminals is performed in this system by constraints on attribute annotation associated with each $\overline{X}$ rule. The set of constraints is partitioned into (non disjoint) subsets that are indexed to each rule, and must be satisfied for the rule to apply. For instance, the following rule is indexed into the set of constraints by the index `postcomplement`.

(16) `x(1,C)==>[x(0,C),x(2,C1),postcomplement].`

The index `postcomplement` selects a subset of the pool of constraints that can apply to $\overline{X}$ rules, some of which are shown in Figure 2, in Prolog-like pseudo code.

```
constraint(+Index,+Head,+Satellite,−MotherNode)

% right compl of preposition
constraint(postcomplement,
           Head,
           Satellite,
           MotherNode) ←

      Head has_category prep,
      Head = MotherNode.

% right complement of adj, n, adv.
constraint(postcomplement,
           Head,
           Satellite,
           MotherNode) ←

      Head is lexical,
      thetamark(Head, Satellite, MotherNode).
```

Figure 2: The constraints indexed to the `postcomplement` rule

Constraints can be divided logically into two groups depending on their function in the parser. Some constraints can affect the way the parser attaches nodes to the already built structure, *i.e.* they can affect the shape of the tree. Other constraints do not: they simply annotate the nodes of a tree already built. Categorial restrictions on the cooccurence of phrases can affect the shape of the phrase marker. On the other hand, morphological feature annotation and feature percolation from the head of the phrase to its maximal projection do not really affect structure-building decisions during the parse. We take advantage of this fact by disassociating feature checking that affects structure-building operations from feature percolation.

On one hand, the separation of rules from constraints has two main advantages. Firstly, it engenders a succinct grammar, because it reduces the number of $\overline{X}$ rules. We obtain the multiplicative effect of several interacting principles, while only using memory resources which correspond to the sum of the sizes of the principles. Moreover, this design achieves language-independence, since it uses highly abstract rules. The same system of constraints, appropriately modified, is used in the Arabic parser as well. For instance, Arabic also has a rich agreement system and the parser uses the same agreement checking mechanism.

On the other hand, since category-neutral rules are used, the parser can generate many structural hypotheses which need to be filtered out. This approach raises the so-called *interleaving* problem, for both Arabic and German: how are the constraints and the phrase structure rules going to be coupled? There are three possible options:

1. All possible phrase structure rules are used, constraints are applied to a forest of trees. This approach is adequate, but it is very space intensive, a forest of hundreds of trees can be built for even small grammars, and it is not very explanatory, in that the entire search space is visited.

[5] Interestingly, an approach where empty categories are postulated as soon as possible, with a partially top-down procedure, has also been used by Crocker (1993). With this design, German and English can be parsed by the same algorithm.

2. All constraints apply at every reduction. This approach is also adequate, but it is not explanatory or efficient, because it applies many constraints in configurations where they are vacuously true.

3. Only a subset of the constraints applies to every rule.

In this system we have adopted the third approach. We have implemented a linking mechanism based on the syntactic configuration. In our system, configurations are partitioned into four main types (following Kornai and Pullum 1990): complement, specifier, adjunct and modifier. Each configuration can appear in two linear orders. A subset of the constraints is indexed to each rule, and must be satisfied for the rule to apply.

While this approach has the problem of not being sufficiently general, since careful tailoring of the interleaving of structure, category and other principles was needed, it is of interest because we found that it eliminates non-determinism more than other approaches. For example, our algorithm to insert empty categories, which relies on structural licensing and theta assignment, is able to eliminate incorrect empty categories as soon as they are postulated, thus it never incurs the explosion, found for example, in principle-based parsers which use functional determination of empty categories (Fong 1991). Also, this approach does not reach the level of specificity that would confine its applicability to only one language. Although we are not able to propose an algorithm to compile automatically the interleaving of principles and rules, we propose a schema that works for such different languages as German and Arabic.

To illustrate, the complete set of $\overline{\text{X}}$ rules and the main constraints indexed into each rule are shown in Table 1. Table 2 shows what features are manipulated by each constraint. Finally, the entire pool of constraints is shown in Table 3.

RULE	CONSTRAINT
prespecifier	categorial selection agreement θ-marking feature percolation
postcomplement	categorial selection lexical/functional selection conjunction of likes θ-marking feature percolation
precomplement	categorial selection θ-marking feature percolation
premodifier	categorial selection feature percolation
postmodifier	categorial selection feature percolation
preadjunct	categorial selection feature percolation
postadjunct	categorial selection feature percolation

Table 1: Interleaving of Rules and Constraints

CONSTRAINT	FEATURES
agree	number of head and sister person of head and sister nominative case for sister
categorial selection	category of head and sister
lexical/ functional	category of node
θ-marking	obligatory case and θ-role or optional case and θ-role θ-grid: θ-roles and case
conjunction of likes	category of conjunct
feature percolation	number of head and sister gender of head and sister person of head and sister *wh*-feature

Table 2: Constraints and their Range of Features

5 Error Handling

Since the parser described so far is to be used for tutoring, one very important module of the system is the error handler which detects and diagnoses mistakes. Error tolerance is important to avoid aggravating situations in which the student interacts with a system which is not sufficiently flexible, since the typical user of this system is likely to produce ill-formed input of several kinds: misspellings, agreement errors, (such as wrong declension for nouns and adjectives or wrong conjugation for verbs), syntactic mistakes, (such as putting words in the wrong order, in Ger-

CONSTRAINT	FUNCTION
agree	• checks case assignment and person and number agreement between verb and subject • percolates intersection of features to mother node
categorial selection	• checks that category of head and sister are compatible • percolates category of head
lexical	• checks membership of node to lexical categories
functional	• checks membership of node to functional categories
θ-marking	• checks availability of θ-role for sister • assigns a θ-role • modifies the θ-grid of the head
conjunction of likes	checks categories of conjuncts
percolate	• checks number, gender, and person features • percolates features to mother node

Table 3: The Constraints

man, *e.g.* failing to put the verb at the end of an embedded clause) and also incorrect choice of words semantically, using for instance a movement verb like *legen* instead of the static verb *liegen* to mean *being located.* The parser must also detect and identify the error, while being able to proceed with the analysis. Because error detection is used to build a model of the student and to determine the sequence of learning activities, the diagnostic messages must be accurate, rather than generic. Accuracy is furthermore crucial in choosing the correct default substitute value for a piece of information, which is missing because the input is incorrect. Finally, the system must be flexible, namely detect and diagnose errors at different levels of restrictiveness.

To meet these criteria we have built an error handling facility which is constituted of three logical components. One component performs error detection and produces default values for the mother node if feature matching fails. Upon reduction of a rule, some constraints must be met. One of the tasks of these constraints is to compute the feature set to be assigned to the node that results from the reduction. For example, upon reducing the rule *prespecifier* shown in (17) a constraint on agreement must be satisfied.

(17) `x(2,C) ==>[x(2,C1),x(1,C),prespecifier].`

The constraint states that when reducing a verb and a subject, they must agree in person and number and the subject is assigned nominative case. If agreement succeeds then the result of the unification is used to annotate the mother node, while the error list is empty. If it does not succeed, then the feature to annotate the mother node are determined by default (usually the features of the head are simply copied), and the error list will contain an error code used to diagnose what kinds of errors have occurred.

A second component of the error handler produces messages, based on the code passed by the parser, and retrieves the lexical items which caused the error, by traversing the subtrees in the tree stack. If the error list contains more than one error code then more than one message is generated.

The third component is a set of control switches, which determine how restrictive the parser is going to be in diagnosing errors and reporting them to the tutoring module. Upon detection of a mistake, the error handler checks whether the switch for that particular kind of error is on; if it is, an error message is produced, otherwise the parse proceeds silently.

The use of constraints which are separate from phrase structure rules is crucial in supporting error tolerance. As discussed above, some of the constraints for feature assignment are not needed to determine the shape of the tree. Thus even if such features are missing, the parse tree can be constructed for semantic analysis. At present the error handler in the German parser can detect the following types of errors:

(18)
- Subject verb agreement
- Noun-adjective agreement
- Case errors
- Complement selection errors

- Preposition selection errors
- Errors in selection of auxiliary by verbs
- Word order errors
- Spelling errors

The Arabic error handler uses the same mechanisms to handle a variety of errors, mainly in involving agreement in nominal phrases, which can be classified as follows:

(19)
- Plural formation errors
- Agreement within *noun construct* errors
- Adjective-noun agreement errors
- Subject-verb agreements errors

6 Evaluation

In this section we assess how the initial design goals of robustness, flexibility, portability, and efficiency have been met.

There are two sides to the definition of how robust the system is. First, how wide the linguistic coverage of the natural language processing system is, and second, how tolerant the system is to incorrect input. We have already discussed the error handling ability in some detail.

As far as coverage is concerned, the German parser handles declarative, imperative, and subordinate clauses, *wh*-questions and yes-no questions, topicalizations, inversions, conjunctions, constructions with multiple verbs and with modals.

The Arabic parser handles simple declarative sentences with differing word orders, imperatives, subjectless sentences, *wh*-questions and yes/no questions, simple relative clauses, noun construct constructions, clitic constructions, simple embeddings, and sentences with unexpressed verbs, which are very common in Arabic, and must be distinguished from fragments.

Table 4 shows the percentages of success and failure of the parser on a batch of 205 test sentences for German, which were designed by a team of educators for foreign language training (which did not include any of the authors.) Batch test suites are being constructed for the Arabic parser. The German parser is already supporting prototype lessons. Both the Arabic and the German error sets, which are incorporated in the test sentences, were influenced by an analysis of the needs and by real errors made by foreign language students at the intermediate level. In both parsers there is a good fit between the errors diagnosed, the constructions handled, and the needs imposed by the tutoring application. In the Appendix we exemplify some constructions that the parser can handle.

Flexibility is a different way of looking at the features that support robustness. Our parsing system is flexible in the sense that it is modular and that some modules (the feature annotation constraints) may or may not be incorporated in the parser. For example, we can support two versions of this NLP system: a restrictive and a permissive version. In the latter, feature agreements may be ignored, thus partly ill-formed sentences can still be produced by the student without penalty, while in the former version all the errors are detected.

Thirdly, the portability of the design is very satisfactory. Because we make use of a modular design and a theory of grammar that encodes universal principles, we believe that many parts of this implementation could be used for other languages. Of course, the stored words would be different, but the same design and indeed entire pieces of software could be simply incorporated in the parser for a new language. Merlo (1992) has kept several features of this design in an LR parser for English. The adaptation of the entire system to two very different language, German and Arabic, is done. Work is underway to adapt the system to Spanish.

The system is very compact: the German lexicon contains 5000 roots and the Arabic lexicon contains 500 roots. Since both languages have very productive morphological system, this corresponds to sufficiently large vocabularies. The German parser amounts to 1632 lines of Prolog code, while the Arabic parser to 1736 lines. The system fits on a PC platform with all the software necessary to run the lesson, *i.e.* the tutoring system and the software for the multi-media interface, which includes some very large audio files. The system is able to provide feed-back to the student at the same speed of a human tutor.[6]

	correct	incorrect	total
input	141.0	64.0	205
parsed	135.0	61.0	196
%	96.4	95.3	95.6

Table 4: Percentage of Successful Parses on Batch Sentences

7 Conclusions

This paper has presented the design and implementation of a parser for German and Arabic, currently used in a tutoring system for computer-aided foreign language training. This is a good application to test the robustness and flexibility of a parsing sytem, since the input is often ill-formed. Moreover, reusability for different languages imposes a portable design. We have illustrated how to adopt a principle-based approach, where linguistic theory is used as directly as possible.

We have discussed the issues related to interaction of principles, recovery of phrase structure using $\overline{X}$ theory, and recovery of long distance dependencies, showing that a principle-based approach provides an interesting experimental answer. We have illustrated the different design choices by several linguistic examples, which cover an interesting range of constructions in German and Arabic.

Acknowledgements

This work was supported by research grant No. 920427-7519 "New Cognitive Technologies to Improve Foreign Language Training" from the Army Research Institute to the University of Maryland and MicroAnalysis and Design, and also by grant No. 15-890071-62 "Applications of AI to Foreign Language Training" to the University of Maryland and SAIC. Thanks also to the Academic Society of the University of Geneva.

[6]Preliminary measures on the two parsers for an uncompiled (interpreted) version on a SUN workstation have given an average speed of 14 words/second for German and 27 words/second for Arabic.

References

Aho, A.V. — J.D. Ullman (1972) *The Theory of Parsing, Translation and Compiling.* Englewood Cliffs, NJ: Prentice-Hall.

Azadegan, S. — J. Martin — P. Merlo — A. Weinberg (forthcoming) *A Government-Binding Parser for Foreign Language Training.* UMIACS-TR, College Park, MD: UMCP.

Berwick, R. (1991) "Principle-Based Parsing". In: Sells P. & S.M. Shieber & T. Wasow (Eds.): *Foundational Issues in Natural Language Processing.* 115–226. Cambridge, MA: MIT Press.

Berwick, R. — A. Weinberg (1984) *The Grammatical Basis of Linguistic Performance.* Cambridge, MA: MIT Press.

Chomsky, N. (1965) *Aspects of the Theory of Syntax.* Cambridge, MA: MIT Press.

Chomsky, N. (1970) "Remarks on Nominalization". In: Jacobs R. & P. Rosenbaum (Eds.): *Readings in Transformational Grammar.* 184–221. Waltham, MA: Ginn & Co.

Chomsky, N. (1973) "Conditions on Transformations". In: Anderson S. & P. Kiparsky (Eds.): *A Festschrift for Morris Halle.* 232–286. New York, NY: Holt, Reinhart and Winston.

Chomsky, N. (1977) "On *Wh*-movement". In: Culicover P. & T. Wasow & A. Akmajian (Eds.): *Formal Syntax.* 71–132. New York, NY: Academic Press.

Chomsky, N. (1981) *Lectures on Government and Binding.* Dordrecht: Foris.

Crocker, M.W. (1993) "Properties of the Principle-Based Sentence Processor". In: *Proceedings of the 15th Annual Cognitive Science Society.* Boulder, CO.

Dorr, B.J. (1987) *UNITRAN: a Principle-based Approach to Machine Translation.* Ms Thesis, MIT. Cambridge, MA.

Dorr, B.J. (1993) *Machine Translation: A View from the Lexicon .* Cambridge, MA: MIT Press.

Fong, S. (1991) *Computational Properties of Principle-based Grammatical Theories.* Ph.D. Dissertation, MIT. Cambridge, MA.

Gazdar G. — E.Klein — G.Pullum — I.Sag (1985) *Generalized Phrase Structure Grammar.* Oxford: Blackwell.

Jackendoff, R. (1977) *X' Syntax: A Study of Phrase Structure.* Cambridge, MA: MIT Press.

Kayne, R. (1984) *Connectedness and Binary Branching.* Dordrecht: Foris.

Kornai, A. — G. Pullum (1990) "The X-bar Theory of Phrase Structure". In: *Language* 66, 24 – 50.

Merlo, P. (1992) *On Modularity and Compilation in a Government and Binding Parser.* Ph.D. Dissertation, University of Maryland at College Park. College Park, MD.

Shieber, S. (1986) *An Introduction to Unification-Based Approaches to Grammar.* Chicago, IL: University of Chicago Press.

Thiersch, C. (1978) *Topics in German Syntax,* Ph.D. Dissertation, MIT. Cambridge, MA.

Tomita, M. (1985) *Efficient Parsing for Natural Language.* Hingham, MA: Kluwer.

Wanner, E. — M. Maratsos (1978) "An ATN Approach to Comprehension" In: Halle M. & J. Bresnan & G.A. Miller (Eds.): *Linguistic Theory and Psychological Reality.* 119–161. Cambridge, MA: MIT Press.

Appendix

Simple Locative

(1) *Ihr liegt im Sueden*
you.pl lie in the.dat South
'You are in the South'

(2) *Der Berg liegt Suedlich der*
the mountain lies south the.gen
Stadt Lauterbach in Hessen
city Lauterbach in Hessen
'The mountain is south of the city of Lauterbach in Essen'

(3) *Yuujadu maa un fii l-mityaahati*
there be a water in the Mityaha
'There is water in Al-Mityaha'

Simple Predicative

(4) *Ihr frau ist klug*
your.pl wife is smart
'Your wife is smart'

(5) *Al-Mityaahatu laysat 9alaa Tariiqin*
Al-Mityaahatu not be on the road
'Al-Mityaha is not on a road'

Simple Transitive

(6) *Ich habe die Antwort gefunden*
I have the answer found
'I have found the answer'

(7) *Man hat einen guten Rundblick in*
one has a good view in
das Tal hinunter
the valley below
'One can have a good view of the valley below'

Simple Intransitive

(8) *Links neben dem Bach befindet*
left near the.dat stream finds
sich die Eisenbahnlinie
itself the railway-track
'To the left of the stream finds itself (lies) the railway-track'

Conjunctive Phrases

(9) *Ich blicke nach links, nach*
I look toward left toward
Norden und nach Osten
north and toward east
'I look to the left, to the North and to the East'

(10) *Sahraawiyyatin wa Hajariyyatun*
desertlike and stony
'desertlike and stony'

Simple Questions

(11) *Wo stehen wir?*
where stand we
'Where are we standing?'

(12) *In welche Richtung waechst die*
in which direction grows the
moderne Stadt Lauterbach?
modern city Lauterbach
'In which direction is the modern city of Lauterbach?'

(13) *Maa huwa 9adaddu T-Turuqi*
What he number the roads
l-mawjuudaati fii d-dibbati
the found in Al Dibdibba
'How many road are found in Al-Dibdibba?'

Imperatives

(14) *Beschreiben Sie die*
describe.pl you.formal the
Umgebung der Stadt
sorroundings the.gen city
'Describe the sorroundings of the city'

(15) *Sifa l-buyuuta fii d-dibdibbati*
Describe the houses in the AlDibdibba
'Describe the houses in Al-Dibdibba'

(16) *da9naa nufakkiru bi mandharin*
we will consider conj land
Tabii9iyyin namuuthajiyyin fii gharbi
nature typical in west
l-bilaadi
the country
'Let's consider a typical landscape in the West of the country'

Modals

(17)

Was	*kann*	*man*	*auf*	*dieser*	*Skizze*
what	can	one	on	these	sketches
von	*Lauterbach*	*erkennen?*			
of	Lauterbach	recognize			

'What can one recognise in these sketches of Lauterbach?'

(18)

Man	*kann*	*ein*	*Burg*	*erkennen*
one	can	a	fort	recognise

'One can recognise a fort'

Inversion

(19)

Oestlich	*von*	*Lauterbach*	*liegt*
east	of	Lauterbach	lies
Angersbach			
Angersbach			

'East of Lauterbach lies Angersbach'

Embedded Constructions

(20)

Ich	*denke,*	*daß*	*Peter*	*und*	*Hans*
I	think	that	Peter	and	Hans
nach	*Deutschland*	*gegangen*	*sind*		
towards	Germany	gone	are		

'I think that Peter and Hans have gone to Germany'