Parsing Scrambling with Path Set: A Graded Grammaticality Approach

Siamak Rezaei (Durroei)
Université du Québec à Montréal
Montréal, Canada
rezaei_s@gip.uqam.ca

Abstract

In this work we introduce the notion of path set for parsing free word order languages. The parsing system uses this notion to parse examples of sentences with scrambling. We show that by using path set, the performance constraints on scrambling such as Resource Limitation Principle (RLP) can be represented easily. Our work contrasts with models based on the notion of immediate dominance rule and binary precedence relations. In our work the precedence relations and word order constraints are defined locally for each clause. Our binary precedence relations are examples of fuzzy relations with weights attached to them. As a result, the word order principles in our approach can be violated and each violation contributes to a lowering of the overall acceptability and grammaticality. The work suggests a robust principle-based approach to parsing ambiguous sentences in verb final languages.

1 Introduction

Numerous formalisms and systems have been designed for representing the grammar of free word order languages [10], [18], [19], [9], [15], [7], [14]. Each formalism has tried to capture some examples of local scrambling or long distance scrambling. Some of the formalisms considered the role of discourse in scrambling and the fact that under a specific intonation, one word order may be more acceptable. Another issue which has not been thoroughly investigated is the notion of *acceptability* itself and implementing this imperfect notion for scrambling cases. Recently, notions such as probability, optimality, possibility, plausibility, acceptability and graded grammaticality have been incorporated into the linguistic theories. Despite the fact that scrambling and word order introduce a degree of acceptability and graded grammaticality, nevertheless the necessary acceptability or plausibility notions have not been added to the scrambling rules.

Modelling graded grammaticality has been neglected in many of the past works, and [2] is one of the few that tried to incorporate them into the linguistic competence. Is graded grammaticality part of competence or performance or part of both?

Graded grammaticality and its interaction with word order constraints have also been studied from another perspective in *performance* models for languages [6]. The main problem is that not many significant theoretical works have been done to incorporate graded grammaticality in a unified model of competence and syntax. The lack of methods for gathering data and formal models of graded grammaticality are also complicating the problem.

For a flexible word order language such as Persian, an Indo-European language spoken in the Middle East, accounting for graded grammaticality is essential because there are different levels of ambiguity in the grammar that interact together. In Persian the subject and object of a sentence can be missing (i.e. Pro-drop property) and subject and object marking is ambiguous in some cases. The notion of *specificity* which is a graded notion in Persian plays an important role in the disambiguation between subject and object. So modelling graded grammaticality becomes essential and it interacts with the word order rules.

In a computational framework, one can model graded grammaticality as a form of competition among a set of alternatives with different degrees of grammaticality. In a competition framework, the result depends on the entities that are taking part. The violation of the principles of the grammar reduces the graded amount of grammaticality (i.e. acceptability) for each alternative. Competition in a grammar can arise for acquiring the highest degree of grammaticality among a set of plausible interpretations, but competition can also arise for limited linguistic *resources*. What are these resources and are there specific principles in languages that put further restrictions for acquiring these resources? We will answer these questions in the specific domain of modelling Persian and the scrambling in its word order.

In this paper we will look at some of these issues and by introducing competition and parallelism at the same time. We avoid some of the problems of backtracking and the inefficiency that it causes. We will further investigate linguistic limitations which one can impose on the processing architecture to restrict some of the possible alternatives. For this purpose we turn to recent proposals for adding resource limitation strategies to the processing [8].

The structure of the paper is the following: in Section 2 we discuss the details of the parser. Section 3 illustrates the parser. In Section 4 we discuss some major aspects of the parser.

2 A Pipeline Parser

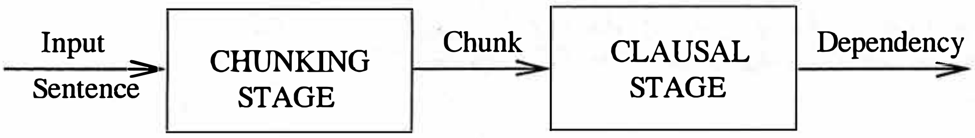

Figure 1: Parser Modules

Based on the grammar of Persian and previous experience in parsing Persian by PATR-II [16] and [17] we have implemented a two level parsing system.

(1) **Main Body:**

PAR (run in parallel)

a. parse-chunk(Pipe) to read a word and output a chunk on the pipe.

b. parse-clause(Pipe) to read a chunk from the pipe and output dependencies.

The first level of the parser, which is a variant to the PATR-II system, groups the words of the sentence into chunks: NP, PP, V, and Comp using context-free phrase structure rules. As soon as a chunk is found it is passed to the second level of the parser. The two stages are run in parallel in contrast to [12] (which the parser is based on). Abney [1] uses a similar notion of pipeline parsing. He refers to the first stage as *chunk level* and to the second stage as the level of *simplex clauses.* Abney uses a finite-state cascade and his system uses finite-state models for grammatical representation at both stages. Instead of finite-state models we have used an extension to CFG rules in the first stage and regular grammars for the second stage. CFGs are more flexible and powerful in representing constituents with levels of recursion. We have also introduced a look ahead for these rules at the first stage.

For representing scrambling we have extended the regular grammar rules for clauses with a special path set that keeps record of possible interpretations for the arguments of the clause. This path set is used to represent competition for grammatical functions and backtracking is avoided. It is updated incrementally. For example if the first constituent can be attached to the clause as SUBJect and OBJect, and if the next constituent can be attached as both SUBJect and OBJect, then the path set will include all possible combinations of: [SUBJ.SUBJ, OBJ.SUBJ, SUBJ.OBJ, OBJ.OBJ]. Some of these possibilities are restricted by the use of *word order constraints.* In this example SUBJ.OBJ is referred to as a *path.* Each path in the path set has an activation or possibility value attached to it which shows the plausibility of that particular path relative to the others. The value corresponding to each path in the path set is calculated based on word order constraints and the numeric values considered for each word order constraint.

In other words, in our framework the word order constraints are defined locally to a clause (and not for rules), and they specify the precedence relations between two grammatical functions. The precedence relations are probabilistic and each possible word order has a probability measure attached to it.

The word order constraints are of two types: hard and soft. The hard constraints cannot be violated, while the soft ones can be violated. The violation of a hard constraint makes the corresponding path inactive, while the violation of a soft constraint reduces the level of activity of that specific path. For simplicity we assume that the activity level is the same as a probability number.

In the following we will explain the details of the system and will elaborate on hard and soft constraints that put restrictions on these alternatives (paths).

2.1 First Stage

For parsing the phrase structure rules of the grammar we have used a Prolog implementation of the standard version of PATR-II.

We will first review a simple example of parsing.

The input sentence is *ali seab xord.*

(2) ali seab xord.
Ali apple ate
'Ali ate an apple.'

1. Dictionary look up:
   Input: `[ali, seab, xord].`

   Output:
   `Noun(ali,3,80), Noun(seab,3,20), Verb(xord,3,`$< Obj, Subj >$`).`

2. phrase chunking: (bottom-up)
   Input: `Noun(ali,3,80), Noun(seab,3,20), Verb(xord,3,<` $Obj, Subj$ `>).`
   Output:

```
NP(dp(ali),3, ▷ obj:20 ◁, ▷ subj:80 ◁)
NP(dp(seab),3, ▷ obj:80 ◁, ▷ subj:20 ◁)
verb(verb(xord),3, < Obj,Subj >, 100)
```

Noun(ali,3,80) gives this information about *Ali* that is 3rd person singular (3) and has a specificity of 80. ▷ gram-func:activation ◁ such as ▷ obj:20 ◁ shows a pair of grammatical functions and activation values. Each constituent (chunk) may have one or more of these pairs. The number indicates the plausibility of that grammatical function for the constituent. The verb entry also shows that the verb has an object and subject and is third person singular (3). We have used an activation value of 100 to raise the activation of clauses that have verb, compared to those which lack one and are not completed. Note that we have assumed no ambiguity for the verb and hence the activation value here reflects the notion of possibilty of this interpretation.

At this stage we specify for each marked NP the possible grammatical functions that it can accept. The numbers after the grammatical functions correspond to the possibility of that alternative. These numbers are derived from the specificity value of a noun and the presence or absence of *ra* after the constituent. For example in the above *Ali* is a proper noun and it is specific. Since it is not marked by *ra* (specificity object marker), its object value is low (20%) and its subject value is high (80%). For NPs which are not marked with *ra* we have considered subjecthood equal to the specificity value and objecthood = 100 – specificity-value. We have used a numeric value for specificity because specificity of a phrase varies over a non-discrete range. In the absence of a corpus for deriving the probabilities of words and their co-occurrence we have used this notion to initialize the activation value, because we mainly use it for subject-object disambiguation which relies on specificity.

In contrast *seab* 'apple' is not a proper noun; as it is not marked by *ra*, it can be either subject or object. For objects like *seab* the subjecthood value of 20% and objecthood of 80% have been considered. This is because the corresponding specificity value for *seab* is 20. Note that one can consider different numbers, but the choice of numbers and their relation with specificity and object marking by *ra* should be taken into account.

Our goal in designing the phrase structure (PS) component of the parser was to parse the input string into chunks and pass these chunks to the next level of parsing. By using the parallelism concept of Linda, the interface between the two stages is implemented. In our model the chunks are transmitted as Linda tuples between the two stages.

2.2 Parsing Stage II

At this stage, the constituents of a clause are assembled. This stage is run in parallel with the first stage and as a chunk is produced in the first stage, the attachment of it to the clause will be started. In other words, the second stage processes the chunks incrementally. The grammatical knowledge at this stage is represented procedurally and the parser gets the incoming chunks (from previous stage) and adds them to the clause that it is currently processing.

Depending on the incoming chunk, there are different cases. The finite-state model of this module is shown in Figure 2.

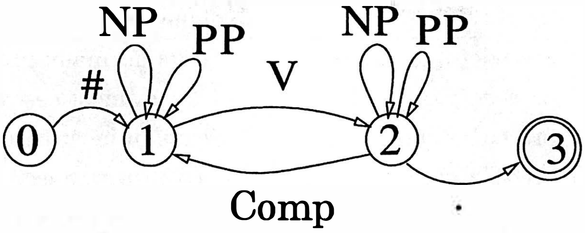

Figure 2: Second Stage

```
(3)     Main Body:
        parse-clause(Pipe)
          a. Initialize a new Clause
          b. Input a Chunk from the pipe.
          c. Do while Chunk not end of sentence.
            i. If Chunk is a complementizer: attach-new-clause(Clause).
            ii. else If Chunk is a phrase: attach(Chunk,Clause).
            Input a new Chunk from the pipe.
```

In the first case (c.i) the chunk will be added to the present clause and the parser continues with reading the next chunk and adding it to the present clause. In the other case (c.ii), the parser spawns a new clause and initialises the variables of the clause with chunks that can be exported into it. In this way the parser represents long distance scrambling and control. When the parser reaches the end of the sentence, the work of the parser is completed.

```
(4)     Main Body:
        attach(Chunk,Clause)
          a. If Chunk is NP:
            i. Add grammatical functions of the Chunk to the Path-set of Clause.
            ii. Use Apply-filter to impose word order constraints on the new Path-set.
            iii. Update Export-set (and Import-set) for long distance scrambling.
          b. If Chunk is PP:
            i. Add the grammatical function of the Chunk to the Clause.
            ii. Block ungrammatical parses which violate the word order Principles.
            iii. Update Export-set (and Import-set) for long distance scrambling.
          c. If Chunk is Verb:
            i. Add the subcategorization frame of the Chunk as expected-resources of the Clause.
```

For representing local scrambling, we have used the notion of the path set. This notion allows one to have competing alternatives of plausible word orders and rank them according to some constraints.

The word order constraints that we have considered are listed in Table 1. The word order constraints are designed to reduce the activity of those alternatives which are not marked and which deviate from the canonical word order. A zero in the precedence constraint imposes a hard constraint to filter out illegal word orders[1]. A non-zero value imposes a soft constraint to reduce the activation value for non-canonical word orders.

[1] We introduced the notion of filtering in the algorithm (4).

Implemented	Constraint	Explanation
Yes	precede(obj,subj, 0.90)	Subjects normally precede objects
Yes	precede(v,obj, 0.20)	Objects in most cases precede verbs
Yes	precede(v,subj, 0.20)	Subjects in most cases precede verbs
No	precede(obj,topic, 0)	Topics always precede objects
No	precede(obj2,obj, 0)	Object always precede object2
No	precede(obj2,subj, 0)	Subjects always precede object2

Table 1: Precedence Constraints in the Second Stage

3 Parsing Local Scrambling

The constituent rules in this stage are simple CFG rules. A clause can be generated as a result of the combination of a clause and a constituent, or it can introduce a new embedded clause, or by reaching the end of sentence, a clause can be terminated.

These automata do not specify the precedence relations between the constituents, and a separate Linear Precedence component imposes the precedence constraints. This is done incrementally and as a constituent is added to a clause, all the possible word order constraints are applied between it and the constituents which are already part of the clause. Note that we haven't considered any immediate dominance (ID) component and the binary precedence relations are not imposed on sisters of an ID rule. The system parses a sentence by initialising a clause and attaches the incoming chunks to this clause.

For (2), repeated in (5), the first chunk is *Ali.* As a result of incremental attachment at this stage we will have:

(5) ali seab xord.
Ali apple ate
'Ali ate an apple.'

np([0,1],▷ obj:20 ◁,▷ subj:80 ◁)
Rule: Clause → NP Clause

Candidates:
▷ [0,1]:obj ◁ 44.72
▷ [0,1]:subj ◁ 89.44

We have kept the indexes for each constituent. For example [0,1] shows that this constituent starts at point 0 and ends at point 1 in the input string. We use these indexes in generating the output dependecies for the parser. The parser generates these after it reaches the end of the clause (not sentence) which it is parsing.

The candidates also show the competing paths in the path set for each clause. At the beginning when the clause is initiated this path set is empty and after parsing the first constituent, the candidates (or path set) will be initiated. It is at this stage that the activation values for each path will be calculated. We have assumed 100 as the initial number for an empty clause and when it is combined separately with 20 and 80, the results will be 44.72 and 89.44. 100 is the maximum value of activation and the activation value can range from 0 to 100. We have used the square root function because it implements blocking constraints as reduction by multiplication to zero.

$\sqrt{20 \times 100} = 44.27$

The second chunk is *seab* and as a result of multiplication, we will have 4 grammatical-function pairs as potential candidates in the path set: subj.obj, obj.subj, obj.obj, subj.subj.

np([1,2],▷ obj:80 ◁,▷ subj:20 ◁)
Rule: Clause → NP Clause
Candidates: | ▷ [0,1]:obj ◁▷ [1,2]:subj ◁ 28.37 |
| ▷ [0,1]:subj ◁▷ [1,2]:obj ◁ 84.58 |

At this stage only two of the four possible alternatives can pass the filters. Since no sentence can have two objects or two subjects, the activation values of those sequences which have two subjects or two objects are reduced to zero and only two will survive.

verb-comp([2,3],$< Obj, Subj >$,100)
Rule: Clause → V-comp Clause
Candidates: | ▷ [0,1]:obj ◁▷ [1,2]:subj ◁ 28.37 |
| ▷ [0,1]:subj ◁▷ [1,2]:obj ◁ 84.58 |

When the verb is added with the activation of 100, those subjects which don't agree with verb will be deleted. Since both of the subjects agree with the verb, both alternatives will survive[2]. Finally, for the attachment of the arguments to the verb the path with the highest activity (acceptability) will be chosen and the arguments are bound to the verb. Since no word order constraint has been violated the activation value will be $91.97 = \sqrt{84.58 \times 100}$.

Note that with the same constituents and a different order, the constraints will interact to yield a different measure of acceptability. For (6) the acceptability measure is 89.58. This is because the example with canonical word order is considered more correct.

(6) seab ali xord.
apple Ali ate
'Ali ate an apple.'

In this example, the object precedes the subject and hence violates the canonical word order. As a result the activation will be multiplied by 0.90 (see Table 1 for precedence rules). A simple matrix for deriving the acceptability measure can be calculated by multiplying the values for the constraints which were violated. One can calculate all violations and yield a final violation measure and multiply the end result with this number. Instead we have multiplied each violation as soon as it is found. This incremental approach ensures that alternatives which are reduced to zero are not further extended.

Finally in Persian, PPs can scramble freely and in parsing them we do not add them to the competition (unmarked) set, because they contribute the same to all the competing paths. Instead of adding them to all paths we factor them out and store them in another marked structure because their contribution to all parallel paths are similar.

The subcat(egorisation) expectations of the verb are also added to a separate structure in our model and when the end of the clause is reached the resources and the expectations are matched with each

[2] For avoiding confusion, we have not shown the agreement features in the examples.

other and the dependency links are generated. In our model we choose the path with the highest activation and discard the other ones.

Note the difference in order of items for subcat resources and the normal resources. In subcat we have subj:[4,5] where [4,5] corresponds to the location of the verb in the sentence, while in the unmarked resources we have [0,1]:subj (note the difference in the order of grammatical relation and the brackets in the two). When the parser reaches the end of a clause, the highest active path in unmarked will be selected and the matching /bf subj resource and subcat /bf subj resource[3] are joined.

As a result of joining these two a dependency link with value [0,1]:subj:[4,5] will be generated. This depicts a transaction or communication across a subject communication link; in this transaction [0,1] is the producer and [4,5] the receiver. Similarly two other transactions for obj and pp(ba) links will be generated by this distributed approach to communication resources.

If some of the resources in marked or unmarked parts could not be unified by a corresponding element in the subcategorization frame of the verb, then these resources are moved into the embedded clauses. This is because of long distance scrambling in Persian in which some resources might belong to other embedded clauses. It is also possible that some of the expectations of the verb might not be satisfied due to the nature of Pro-drop in Persian for the arguments of a verb[4]. The feature EXT(ernal)-COM(munication) is introduced for these cases and is a set based extension to traditional gapping for long distance scrambling.

To sum up, in parsing local scrambling, as the unmarked arguments (subject, object) are added incrementally to the clause, the parser creates a parallel set of the plausible paths. The paths are restricted by some constraints. The constraints on local scrambling can be divided into hard and soft constraints. The hard constraints are strict precedence relations and verb-subject agreement which could block a path by reducing its activation value to zero. The other constraints which only reduce or increase the activation values to a non-zero value are soft constraints. The accumulative result of these values contributes to the possibility (activation) of a solution. The most active solution or path (i.e. among unmarked paths) will be chosen. Depending on the function which we use we will have different results. The constraints can be summerised as:

- Word order restrictions. These were illustrated in 1 and are used to penalize possible alternatives which deviate from the canonical word order. They also block alternatives which violate obligatory word order rules.
- Verb subject agreement. In Persian a subject must agree with the verb of the clause.
- One example of each resource in the sentence or Resource Limitation Principle (RLP).

(7) **Resource Limitation Principle**
No two NPs can exist in a clause with the same grammatical function.

In the rest of this section we will elaborate on RLP. Consider example (8).

(8) amir seab be man qol=dad [ke be ali bedahad].
Amir apple to me promise=gave-3S that to Ali gave-3S
'Amir promised me to give the apple to Ali.'

[3] In our model we consider grammatical relations as pairs of links that attach NPs and Verbs. These links can be considered as communication resources between two linguistic processes [17].

[4] The number of fulfilled expectations can contribute to the activation positively, but we have not considered it in our implementation.

In Persian it is not possible for two grammatical resources with the same grammatical function to appear in the same clause. Hence (9) is ungrammatical.

(9) * amir seab be man be ali qol=dad [ke – bedahad].
Amir apple to me to Ali promise=gave-3S that – gave-3S
'Amir promised me to give the apple to Ali.'

In other free word order languages such as German, such an example with two dative NPs does not create a problem.

This performance constraint that we call Resource Limitation Principle (RLP) is not restricted in Persian to datives, and no clause can exist in which two phrases (resources) have the same grammatical function. RLP has been implemented in our system as a general constraint that a resource cannot precede another with the same case or grammatical function (as is implemented in our system). We have used an extension to blocking word order restrictions. For example the constraint that 'no subject can precede another subject' implements the existence of at most one subject-marked resource in a clause.

4 Discussion

The path set implements a mechanism for modelling competition among a set of paths. Corresponding to each path we had an activation value. By assigning a value to each path, we implemented a numerical notion for competition. Such competition notion is robust enough to capture soft and hard constraints for word order rules. Each word order rule had numeric value attached to it. In modelling competition numerically, one can easily represent blocking as reduction of the activation value to zero. Another advantage is that such a mechanism can be extended to allow robustness and degrees of ungrammaticality. In the following we will discuss major aspects of our work in comparison to other approaches.

Comparison With Classical Word Order Rules

ID/LP [3] can be considered as the classical approach for representing flexible word order. ID/LP uses a set of immediate dominance (ID) rules and a distinct component for linear precedence (LP) which specifies the precedence relations between the right-hand side sisters in ID rules.

Unlike a phrase structure (PS) rule which specifies two distinct relations of ID and LP at the same time, the order of the constituents in an ID rule is specified separately by LP component and in this way ID/LP format captures word order generalisations. The advantages arising from factoring out the ordering component from constituency rules are particularly evident in the case of languages with a flexible word order.

The linear precedence relations in LP component are binary relations and they can only be specified for two sister categories in the right hand side of an ID rule. As a result, no precedence relation can be specified for two categories which do not occur as sisters of a single ID rule.

Another restriction of classical ID/LP rule is the prohibition against referring to the categories inside the internal structure of phrases, in the LP relations. In other words, the LP relations can only specify relations between two sisters in an ID rule and not relations between one sister and another category dominated by the other sister.

In our approach we haven't employed ID rules and instead have used regular rules which allow different word orders. The possible word orders are restricted by a separate notion of word order binary constraints which restrict the possible order of grammatical relations in the paths.

Reape introduces word order domains to deal with word order in Germanic [15]. In his approach, the word order domains of the constituents that join with each other are merged. Unlike the word order domain in Reape's notation, in our approach we only allow one instance of a grammatical resource to be present in a word order path in each domain and each word order domain can consist of a set of parallel competing word order paths. Corresponding to each possible word order path in the word order domain, we have an activation measure.

The activation measure for a specific word order path is reduced if a word order constraint is violated. The reduction corresponds to the strength of that constraint and the stronger the constraint, the bigger the reduction. In the case of hard constraints, violation of a constraint makes the word order illegal and blocks that word order path.

In this way relaxation of word order constraints can be achieved. A general restriction of classical LP constraints is that they cannot be relaxed and they must always be satisfied. [18] proposes to extend these by use of complex LP constraints. In complex LP constraints, as long as at least one of the LP rules is satisfied the other LP rules can be violated. Our approach is a numerical extension to LP rules which allows the possibility of relaxing the LP rules. Introducing a complex LP extension is also feasible in the model to have non-binary word order constraints.

Similar to fuzzy logic sets, one can consider linguistic measures for referring to different relaxation possibilities for each word order rule and a linguist can use these for encoding the strength of the word order rules. Ideally, the relaxation of word order relations and their strengths should be derived from a corpus of texts, so that the most dominant word order gets the highest activation. In other words, these word order rules are statistically prevalent and are designed in such a way that the less plausible word orders get penalized and their activation gets reduced.

In our framework we have used the unmarked order as the most optimal path and deviations from this unmarked order are penalized. This approach can be considered as an extension to a notion of optimal parsing introduced in [6] based on typological research.

Comparison With Other Approaches

Another alternative to modelling the competition and word order constraints is to use Optmality Theory (OT). OT [13] uses a notion of constraint ranking. For parsing free word order languages the cumulative sum of constraints from Syntax, Semantics, Discourse and world knowledge determines the grammaticality of an utterance and the preference of one alternative over another. It is not clear whether one can come up with a constraint ranking of these separate modules.

[11] has also shown that for implementation of OT in a finite-state framework, one should restrict the OT model and a subset of it be considered. Another alternative is to use a cumulative and weighted approach to ranking the constraints[5] that we have adopted.

A general criticism to Optimality Theory (OT) is that the ranking of constraints does not allow any cumulative effect in which a number of lower ranked constraints can compete against a higher ranking constraint. This so-called 'ganging up' effect can be represented by using a numerical representation. OT is a limited case of such a numerical representation with no 'ganging up' effect.

[5] See [4] for another discussion on optimality ranking vs. weighted constraints.

A further criticism to OT parsing model has been raised in the literature [5]. In OT model of parsing, only the most harmonic alternative will be selected and the algorithm does not allow for a number of alternatives with a lower degree of harmony to compete in parallel with the most harmonic alternative, so that if the most harmonic alternative fails, one of the alternatives with lower degree of harmony be selected for the parse to continue. In our work, we have adopted a parallel competitive model that allows a number of alternatives to be run in parallel. This also allows a degree of robustness to be incorporated into the parser in the future.

	PERSIS85	R. Ghasem91	Rezaei92	Rezaei93	Riazati97	SHIRAZ	system
Approach	Production Rule	Conceptual Dependency	ATN	ID/LP	KIMMO PATR	feature str/type	PATR path/LP
Parser	Bottom-Up (BUP)	Procedural	BUP Top Down	BUP	BUP	BUP	BUP
Tokenization	NO	NO	NO	NO	NO	YES	NO
Morphology	NO	NO	NO	NO	YES	YES	NO
Explicit Ezafe	YES	YES	YES	YES	YES	NO	YES
Coordination	YES	NO	YES	NO	NO	YES	NO
Local Scram.	V-final	unrestricted	YES	YES	Limited	V-final	YES
Complement Cl.	YES	NO	NO	YES	NO	NO?	YES
Relative Cl.	YES	NO	NO	YES	NO	YES	NO
Long Dis. Scram.	NO	NO	NO	Fronting	NO	NO	YES
Control	NO	NO	NO	NO	NO	NO	YES
Multiple Parses	NO	NO	NO	YES	YES	YES	NO

Table 2: Comparison and Evaluation

In Table 2 we have contrasted the implemented system with the previous systems for parsing Persian. The system is developed with the goal of complementing the capabilities of the previous systems and it has its limitations. For further details of the above systems see [17].

5 Conclusion

The framework that we discussed in this paper provides a method for adding the notion of graded grammaticality to the principles of the grammar. In traditional approaches to principle based approaches a principle can be satisfied or violated. In our view some of the principles of the grammar can be violated, but the overall relaxation of the principles (when added together) should not reduce the acceptability of the solution below a certain level.

The acceptability of a particular solution is reduced by a factor whenever a principle of the grammar is violated. This factor depends on the contribution and importance of that specific principle.

By adding features to the word order rules, we can introduce more complex word order rules to take into account features such as animacy.

In our work we have introduced performance constraints for scrambling and we discussed the Resource Limitation Principle (RLP) for local scrambling. There is another counterpart to RLP for long distance scrambling - the Resource Barrier Principle (RBP) - that we have also implemented. These performance constraints restrict the scrambling in some free word order languages. They can be used to classify free word order languages.

References

[1] Steven Abney. Partial Parsing via Finite-State Cascades. In *Proceedings of the Workshop on Robust Parsing at Eighth Summer School in Logic, Language and Information*, pages 8–15, August 1996.

[2] Noam Chomsky. Degrees of Grammaticalness. In *Jerry A. Fodor and Jerrold J. Karz (eds), The Structure of Language: Readings in the Philosophy of Language*, pages 384–389. Printice Hall, 1964.

[3] Gerald Gazdar, Ewan Klein, Geoff Pullum, and Ivan Sag. *Generalized Phrase Structure Grammar.* Harvard University Press, Cambridge,Massachusetts, 1985.

[4] E. Gibson and Brienhier. Optimality Theory and Human Sentence Processing. MIT Working Papers in Linguistics (In press), 1998.

[5] Mark Hale and Charles Reiss. What an OT parser tells us about the initial state of the grammar. In *Proceedings of the GALA'97 conference on Language Acquisition*, pages 352–357, Edinburgh, UK, April 1997.

[6] John A. Hawkins. A Parsing Theory of Word Order Universals. *Linguistic Inquiry*, 21(2):223–264, 1990.

[7] Beryl Hoffman. *The Computational Analysis of the Syntax and Interpretation of Free Word Order in Turkish.* PhD thesis, Dept. of Computer and Information Science, University of Pennsylvania, 1995.

[8] Mark Johnson. Resource Sensitivity in Grammar and Processing. In *The Ninth Annual CUNY Conference on Human Sentence Processing*, New York, 1996.

[9] Ronald M. Kaplan and Annie Zaenen. Long-Distance Dependencies, Constituent Structure, and Functional Uncertainty. In *Mark R. Baltin; Anthony S. Kroch (eds) Alternative Conceptions of Phrase Structure*, Chicago and London, 1989. The University of Chicago Press.

[10] L. Karttunen and M. Kay. Parsing in a Free Word Order Language. In *D. Dowty, L. Karttunen and A. Zwicky (eds.), Natural Language Parsing, Psychological, Computational, and Theoretical Perspective*, pages 279–306, Cambridge University Press, Cambridge, 1985.

[11] Lauri Karttunen. The Proper Treatment of Optimality in Computational Phonology. In *Proceedings of International Workshop on Finite-state Methods in Natural Language Processing (FSMNLP'98)*, Bilkent University, 1998.

[12] Michael B. Kashket. Parsing a Free-Word Order Language: Warlpiri. In *24th proceedings of the ACL*, pages 60–66, 1986.

[13] Alan Prince and Paul Smolensky. Optimality Theory: Constraint Interaction in Generative Grammar. Technical Report RuCCS Technical Report #2, Centere for Coginitve Science, Rutgers University, October 1993.

[14] Owen Rambow and Aravind K. Joshi. A Processing Model for Free Word-Order Languages. In *Perspectives On Sentence Processing, edited by Charels Clifton, Lyn Frazier and Keith Rayner*, pages 267–301, Lawrence Erlbaum Associates, New Jersey, 1994.

[15] Mike Reape. Getting Things in Order. In *Discontinuous Constituency, Harry Bunt and Arthur van Horck (eds.)*, pages 209–253, Mouton de Gruyter, 1996.

[16] Siamak Rezaei and Matthew Crocker. A Distributed Architecture for Parsing Persian. In *Proceedings of ICSCS*, Sharif Univ. of Technology, Tehran, December 1995.

[17] Siamak Rezaei-Durroei. *Linguistic and Computational Analysis of Word Order and Scrambling in Persian.* PhD thesis, Division of Informatics, University of Edinburgh, 2000.

[18] Hans Uszkoreit. Constraints on Order. Technical Report 364, SRI, October 1985.

[19] Arnold M. Zwicky. Concatenation and Liberation. In *Papers from the 22nd Regional Meeting of the Chicago Linguistic Society*, pages 65–74, 1986.