

# Relational Grammar in Computational Psycholinguistics

Pavel Vodenski

May 2009

## 1 Relational Grammar

### 1.1 A brief Introduction to Relational Grammar

Relational Grammar (RG) is a formalism that treats grammatical relations as first-class primitives. Whereas in the then-competing transformational (TG) accounts of grammar relations were seen as derivable from structure, RG does not consider structural properties of sentences or attempt to give a complete account of surface structure. Where TG sees a sentence as having a deep structure and surface structure with intermediate states between which transformations occur, RG makes reference to initial and final strata and intermediate strata which capture revaluations of grammatical roles. A major difference between the two is that the rules that determine whether an utterance is well-formed or not in RG have ‘access’ to all strata at once.

RG uses a numbering system to describe the relationship between predicates and arguments; the numbers have theory-neutral correlates<sup>12</sup>.

Stratal diagrams parallel derivations in TG. Take the following example of diagrams of active, passive, and unaccusative constructions:

	P	1	2
Active:	eat	crocodile	woman

---

<sup>1</sup>The numbers in parentheses are not standard Relational Grammar numberings; rather, these come from Stratified Feature Grammar, to which we will return in Section 2.1.

<sup>2</sup>The Chômeur relation does not have a clear correlate in other formalisms; during the course of a Relational Grammar derivation, some terms are ‘forced out’ of their role by others, and are thus relegated into a Chô role (put *en chômage*, as in the French term for unemployment). We will likely omit the circumflex when referring to cho in the rest of the paper.

	1	subject
	2	direct object
	3	indirect object
(7) Obl	obliques (instrumentatives, locatives, etc.)	
(8) Chô	chômeur	

Figure 1: Roles in Relational Grammar

	P	1	2
Passive:	P	Cho	1
	eat	crocodile	woman
		P	2
Unaccusative:	P	1	
	sweat	Frank	

## 1.2 The need for a multistratal approach: the passive

In [12], Perlmutter gives two arguments for a multistratal rather than monostratal or pseudo-bistratal account of the passive.

### 1.2.1 Achinese Agreement

‘The agreement affix on the verb of a clause  $b$  is determined by the *initial 1 of  $b$* .’ Perlmutter argues from parsimony that this is a preferable statement over the monostratal statement: ‘The agreement affix on the verb of an active clause is determined by the 1 [subject], and on that of a Passive clause by the nominal heading a  $\text{GR}_x$  arc.’ This disjunctive statement of the rule fails to capture the generalization that the first rule captures. Monostratal theories can avoid the issue and say that the verb agrees with the ‘Agent,’ but that is simply not always the case.

### 1.2.2 Russian Reflexives

Monostratal theories also fail to explain phenomena like the Russian condition on antecedents of reflexives. We can state a condition in RG that refers (implicitly) to all strata at once: ‘Only a nominal heading a 1-arc can serve as an antecedent of a reflexive.’ In so doing, we provide a unified explanation for diverse types of utterances. In a monostratal framework, attempts based on subject-hood, semantic roles, and thematic notions all fail. At the time of writing of [12], a coherent analysis in a monostratal framework was not available.

### 1.3 What RG is not

Relational Grammar does not generally attempt to describe every aspect of the surface structure form of utterances in a language. It may consider terms that do not directly participate in predicate-argument relationships as evidence of the nature of those relationships. In [4] and [11], for example, the distribution of the Italian reflexive marker *se* is used as evidence that auxiliary selection depends on the presence of multi-attachment of a 1 and 2 arc to a nominal. It is equivalent to speak of RG relations in terms of arcs and roles; an arc represents the sequence of roles held by an argument across all of the strata of an RG stratal diagram—in the simplified tabular representations, it is a column. A term can head more than one arc at a time, which is represented in tabular form by a cell with multiple numbers separated by commas. The Italian translation of ‘Ugo defended himself’ is therefore represented as:

1,2	P
1	P
Ugo	difendere
<i>Ugo</i>	<i>si è difeso.</i>

The *1,2* cell in the above diagram represents the linguistic intuition that Ugo is both the subject and object of defend, in some sense (and the Agent and Patient). RG makes no statement about which instance of Ugo is dominated by the verb and does not try to account (at least in this case) for verb agreement; it will not predict the stress over the sentence, or the constituency of any substring of it. All it does is explain why in:

- (23) a. *Ugo ha difeso Pippo.*  
b. *Ugo si è difeso.*

The auxiliary is *avere* in (a) and *essere* in (b). Note that this unified statement about the distribution auxiliaries and *se* is stated disjunctively in [2, 10, 14] [4, pg 63].

### 1.4 In Comparison to other frameworks

Just like any other grammar framework, Relational Grammar is an attempt to describe in some formal way some aspect(s) of human language. Of course, similar ideas are likely to arise when we attempt to account for the same phenomena—even using different frameworks. There is clearly something similar about grammatical roles and relations as described in RG and

theta roles in other frameworks. Theta roles such as *agent*, *patient*, *theme*, *experiencer*, *goal*, *direction* have been used to describe the way in which predicates relate to their arguments. This approach, however, is lacking for two reasons: (1) it is difficult or impossible to apply unambiguously, and (2) it leads to a proliferation of theta roles. Consider sentences such as “The circle surrounds the square,” or “Acme multiplied its workforce by three.” For these reasons, the simplified account used by RG has been incorporated into other frameworks such as Lexical Functional Grammar, Head-Driven Phrase Structure Grammar, corpora such as the Penn Treebank II and the PropBank. See [9] for a description of a more recent foray, Grammatical and Logical Argument Representation Framework (GLARF) by Adam Meyers and others of the Proteus Project at NYU.

## 2 Unification-based Parsing of Relational Grammar

### 2.1 Stratified Feature Grammar

Larry Moss and David E. Johnson created a unification-ready formalization of Relational Grammar known as Stratified Feature Grammar (SFG) [6, 7]. An analysis of an English ditransitive in SFG is represented as the *stratified feature graph* (*S-graph*) in Figure 2.

$$\left[ \begin{array}{ll} [\text{CAT}] & \text{S} \\ [1] & \text{John} \\ [\text{H}] & \text{gave} \\ [3,2] & \text{Mary} \\ [2,8] & \text{tea} \end{array} \right]$$

Figure 2: S-graph in SRG

Which corresponds to the following stratal diagram:

1	P	3	2
1	P	2	Cho
John	gave	Mary	tea

And the following statement in SFG logic:

$$R1 := [H] : gave \wedge [1] : Joe \wedge [3, 2] : Mary \wedge [2, 8] : tea$$

The two linguistic facts relevant to R1 can also be stated in SFG logic:

$$\begin{aligned} \textit{Ditransitive} &:= [H] : \textit{gave} \wedge [1] : T \wedge [2] : T \wedge [3] : T; \\ \textit{Dative} &:= (3, \underline{2}) : T \ \& \ (2, \underline{8}) : T; \end{aligned}$$

The  $\wedge$  in the above examples indicates usual boolean conjunction. The  $\&$ , on the other hand, is specific to SFG. It indicates that the *Dative* is an extension formula which allows arcs with labels 3 and 2 to be “extended” to (3,2) and (2,8), respectively. These extension formulas are the main workhorse for capturing RG analyses and generalizations in SFG. Facts such as *Dative* and *Ditransitive* can be compiled into S-graphs and multiplied over their lexical anchors to form a lexicalized grammar [6, pg 98].

SFG defines unification over the labels representing arcs (the bracketed/parenthesized lists of numbered relations, or *R-signs*) using the concepts *open*, *closed*, and *extension*. Labels that are bracketed on both sides, such as [3,2,2,1], are totally closed. Labels that are parenthesized on both sides, such as (3,2,2,1), are totally open. Labels parenthesized on one side and bracketed on the other are partially open (or closed) on the right (or left): [3,2) is partially open on the right (which is equivalent to saying it is partially closed on the left). *Extension* is allowed along open sides:

$$(3) \sqsubseteq (3, 2) \sqsubseteq [3, 2, 1) \sqsubseteq [3, 2, 1]$$

Above, a label  $l$  *subsumes* ( $\sqsubseteq$ ) a label  $r$  where  $r$  is more specific than  $l$ ;  $l$  provides a partial description of  $r$ . The ordering of the numbers in the list corresponds to the “depth” of the role in the derivation, which is actually the height of the role in the stratal diagram: a term with label [3,2,1] bears a 3-role in its initial stratum and a 1 role in its final stratum.

There is a distinguished R-sign, 0, a member of a distinguished set of R-signs *Null*. Null R-signs can only occur next to brackets. Null allows us to define *predicate-argument (PA) graphs* and *surface graphs*, which correspond to the notions of deep and surface structure, respectively. Given an S-graph such as is pictured in Figure 2.1, the graph in Figure 4 is the corresponding PA graph and the graph in Figure 5 is the corresponding surface graph. The PA graph contains all of the arcs that do not *begin* with a Null R-sign. The surface graph contains all of the arcs that do not *end* with a Null R sign.

These graphs capture the intuition that John is a logical argument of ill, but that the utterance ‘John seemed ill’ has John as an argument of seem. Do not be intimidated by the proliferation of graph types—the S-Graph is merely the entire stratal diagram, while the PA graph is its first stratum and the surface graph its final stratum. Notice that the surface graph is a

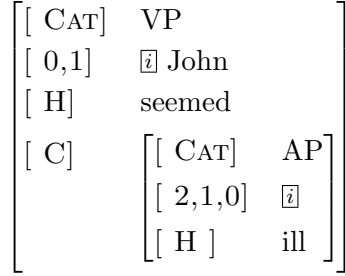


Figure 3: S-graph for *John seemed ill*.

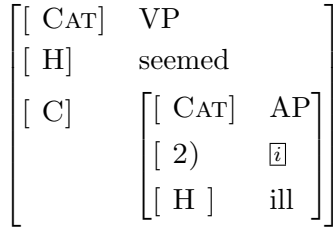


Figure 4: PA graph for *John seemed ill*.

tree; this is an important characteristic of surface graphs and is analogous to the “context-free backbone” of other unification-based formalisms.

We must also define the concept of label unification. Label unification combines two labels to yield a new label with *maximal non-empty overlap*. The result of compiling a lexicalized grammar is stipulated in SFG to be graphs containing labels that are at least partially closed. Therefore, there are four types of label unification during parsing:

1.  $[\alpha] \sqcup [\alpha] = [\alpha]$
2.  $[\alpha) \sqcup [\alpha\beta] = [\alpha\beta]$
3.  $(\alpha] \sqcup [\beta\alpha] = [\beta\alpha]$
4.  $[\alpha\beta) \sqcup (\beta\gamma] = [\alpha\beta\gamma]$

Above,  $\beta$  is the longest common, nonempty string.

## 2.2 An SFG Parser

The parser as presented in [6] is inspired by the head-driven active chart parsing algorithm used to parse Head-Driven Phrase Structure Grammar [13].

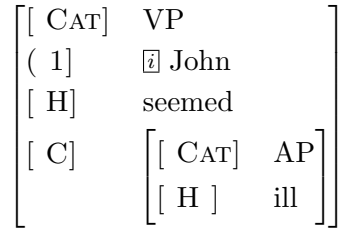


Figure 5: Surface graph for *John seemed ill*.

Given a string of words  $w_1, \dots, w_k$ , it populates a one-dimensional array of state-sets of length  $k$ , each one containing all of the states of the form  $[n_i, c - 1, c]$  as  $c$  ranges over  $1 \dots k$  where  $n_i$  is an S-graph associated with a given lexical anchor  $w$ . There may be multiple  $n_i$  associated with  $w$ ; *raced*, for example, would lead to a state-set containing two states, both with the same  $c - 1$  and  $c$  (left and right boundaries), but with one  $n_i$  for the passive participle and one for the past participle.

After that initialization step, the algorithm makes another pass over the string, again iterating  $c$  over  $1 \dots k$ , attempting for each set in each state-set  $S_c$  to unify each its graph  $n_i$  with some graph in some set in some state-set first to the left of  $S_c$ ,  $S_L$  such that arcs (used here in the unification sense of arc, and not the Relational Grammar sense) within the graph of that state can unify. We have already defined unification for lists of R-signs; unification of other arcs attempts to combine graphs such as in states 2:2 and 3:1 in 6 by combining arcs with like labels and targets, such as the two arcs headed by AP targeting NP. Since unification of their role lists works—it is a case of the third type of label-list unification described above—these two arcs unify and the graphs combine to complete the parse of *John seemed ill*.

In parsing  ${}_0\text{John}_1\text{seemed}_2\text{ill}_3$  (the chart of this parse is provided as Figure 6), after initialization, the parser tries to unify the states in 1:1 and 2:1 and successfully adds the graph in 2:2 to state-set  $S_2$ . Then it attempts to unify 3:1 with 2:2 and succeeds, adding 3:2, which spans the entire string and is complete, making it a successful parse.

### 2.3 Implications of the Parser by Johnson et al

Johnson et al make several assumptions that are not absolutely necessary for using Relational Grammar in computational psycholinguistics. First, they adopt the LFG strategy of pre-compiling all of the allowed variations of a predicate into the lexicon. In other words, rules such as *Dative* and *Ditrans-*

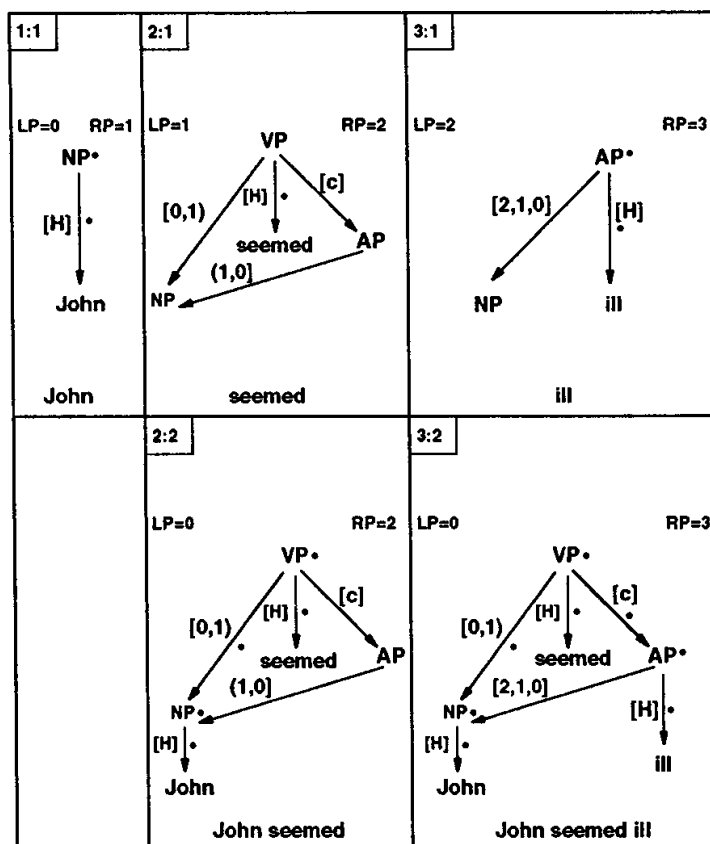


Figure 6: Chart for *John seemed ill*. Extracted from [6, pg 103]

*sitive* as mentioned earlier, as well as—presumably—the rules for passive, antipassive, retroherent unaccusative, etc. are multiplied out such that, if a lexical anchor can unify with three of them freely (to form ditransitive, dative, passive, ditransitive + dative, ditransitive + passive, dative + passive, ditransitive + dative + passive), it will have eight distinct entries in the lexicon (the seven listed and its simple active form). In general, a predicate that combines with any number of  $k$  such rules (including none of them) will contribute  $2^k$  entries to the lexicon. This may not be desirable from a performance viewpoint, especially since one of our goals in making use of RG is to enable true *broad* coverage. It is also not psychologically-realistic. Some such rules are completely productive over all verbs in a language—for example, English *-ing* gerunds; we should not be bound to double the num-

ber of verbs in our lexicon when on-line application of an unconstrained rule seems much more realistic. Of course, it is not wise to assume that it *is* more realistic; experiments that discount the Derivational Theory of Complexity could be interpreted to discount the on-line model of rule-anchor unification.

Moreover, the head-driven chart parsing algorithm as Johnson et al describe it is not incremental. For one, it makes a complete pass over the sentence to initialize the chart *before* performing any meaningful work. During the main run of the parser, the algorithm directs it to iterate over the words of the sentence and, for each word, attempt unification with any graphs already built to its left, and then to its right<sup>3</sup>. This strategy will enumerate all of the possible interpretations of the substrings of the given sentence, which may be desirable from a pragmatic perspective but is clearly not psychologically realistic.

We should not be especially surprised by this, however, since Johnson et al make no claims about the psychological reality of their parser. It is simply a system to do Relational Grammar analyses by computer. Luckily, there are many more possibilities for parsing using Relational Grammar analyses and primitives.

### 3 Psycholinguistic Concerns and Relational Grammar

#### 3.1 A Dream-Machine Parser

Now that we have seen how to formalize RG derivations as unification, we can move past Johnson et al's first pass at a parser for RG and imagine instead our ideal RG parser. For one, we should make use of the most psychologically realistic parsing strategy we can. Rather than be forced to choose between top-down and bottom-up parsing, we can make use of a left-corner strategy. In left-corner parsing using a context-free grammar, the parser starts with a bottom-up step by consuming one word of the sentence and considering all of its possible categories. It then takes a top-down step by predicting one or some or all of the possible categories that can dominate the already-seen symbol. When there are no more top-down steps to be taken, the parser consumes another symbol bottom-up and proceeds. Attempts to prove that human sentence processing is either completely bottom-up or either completely top-down have revealed that a strategy combining the two

---

<sup>3</sup>Alberto Lavelli and Giorgio Satta inspired Johnson et al; they use bidirectional parsing on lexicalized Tree-Adjoining Grammars.

is more plausible.

As we mentioned in Section 1.3, Relational Grammar does not provide (and does not seek to provide) an account of the linear order of terms in a sentence—only the way that predicates relate to their arguments. In order for our dream parser to be useful as a simulator of sentence processing, it cannot afford to make the same omission. One framework that accounts for linear order of terms in natural language in a constrained and principled way is Tree-Adjoining Grammar (TAG). Like RG, Tree-Adjoining Grammar also enriches the lexicon with more complex entries, and, like RG, TAG lends itself to unification-based parsing. In fact, Johnson et al’s algorithm for parsing SFG was inspired by Schabes’ lexicalized TAG methods. TAG has already been shown to account for psycholinguistic processing data of crossed and nested dependencies; it is quite appealing. So, what we seek is a left-corner unification parser for a lexicon whose entries combine TAG elementary trees and SFG graphs. It is encouraging that left-corner parsers have been described for unification [16] and TAG [3].

### 3.2 Relational Grammar Unification as a monotonic model of interpretation

Transformational models of sentence processing have a serious limitation in the realm of psychological plausibility: they are destructive. In a transformational account of processing the sentence *Phyllis broke the vase*, *Phyllis* can be incorporated into a deep-structure tree at subject position, *break* can fill in the role of the verb dominated by *Phyllis* and *the vase* can neatly slide in as its object. The logical structure of the sentence is built up piece by piece as symbols of the string are consumed, with each piece adding increments of information to the understanding of the sentence. Models that display this property, which has been called *monotonicity*, are a subset of the class of *representation-preserving (RP)* models [15].

Representation preservation is a desirable property of psychologically realistic models. The monotonic class of RP models is not a novel idea; Bresnan cited it as a desirable property exhibited in LFG parsing [1]. She found it to be equivalent to the order-free composition property of sentence processing, wherein fragments such as *... unlikely to justify ...* must be interpretable on their own in the same way as they are as part of a full sentence like *John’s actions are unlikely to justify yours*. Her work and early work on D-theory [8] led to active research in RP models. The constraint that establishes a monotonic model can be put simply as “no material can be deleted or overwritten in the description. This means that the description

can be seen as *monotonically increasing set*, where the description at each state will be a subset of descriptions at subsequent states” [15, pg 299].

A unification-based parser that builds RG analyses such as the one described earlier for SFG is truly a monotonic model of sentence processing. First let us pause and consider another sentence parsed in a transformational model, where grammatical roles are read off of the structural relationships between terms at deep structure. In the course of parsing *The vase broke*, the parser recognizes a constituent *The vase* and assumes it to be the deep structure subject of a yet-unseen verb. This is not the only way to imagine parsing such a sentence in a transformational framework. We could also assume that interpretation is incremental at a larger grain-size, which would mean that the parser does not posit any structure for the sentence until it sees *broke*. This would just be evading the problem: since the sequence of terms seen by the parser is *NP VP* in parsing both *John broke...* and *The vase broke...*, it would assign the same deep-structure or thematic role assignment structure or other logical description of predicate-argument structure to both. We cannot allow the parser to wait any longer than that—if it needs to see both the deep subject and deep object of *broke* before it posits any deep-structure at all, it is not incremental in the sense that we need it to be and is *a priori* not psychologically plausible. We could also try to get around the issue by appealing to the semantics available to the parser as it consumes symbols: *the vase* is a semantically unlikely agent of breaking, and therefore the parser should be allowed to wager that it will not occupy the deep-subject position of *break*. This line of reasoning is fine, except when we consider pairs such as *The hammer broke* and *The hammer broke the glass*. In such a case, the transformationally-inclined parser has no choice but to posit one PA structure describing how the argument *The hammer* relates to the predicate *broke* and then retracting or revising that structure when it does or does not see *the glass*.

Consider instead how a unification parser fares on the same examples. Upon seeing *The hammer*, the parser is in a state similar to 1:1 in Figure 6. There is no PA structure associated with *The hammer* as the parser waits to see a qualifying predicate. Upon seeing *broke*, the parser can assign an arc from *VP broke* to *NP The hammer* with a partially open (1) label indicating that *The hammer* is the surface subject of *broke* but that it cannot yet ‘seal’ its initial relation. Here the exact nature of lexical entries in our imagined parser becomes partially relevant: if there is only one entry for *break* that combines with other S-graphs to form its unaccusative and transitive forms, then it will unify with one or the other on-line depending on whether the utterance ends at *broke* or continues with *the glass*. If

both forms are already compiled into the lexicon as separate entries, then the parser has to either enter two states at once (entertain two analyses at once) or create a stack of possible analyses based on some likelihood heuristic, or do something else to account for the ambiguity. The chosen implementation does not really bear on the issue at hand here, but the most fitting choice would actually be a lexical entry for *break* with an optional [1] arc relating it to the entity that causes the breaking. Johnson et al omit a full description of optional arcs from their article *JohnsonEA93*, but indicate that their “parser considers two paths, skipping the optional arc on one and attempting to complete it on the other.” In any case, whatever the strategy for ambiguity resolution, when the parser sees *the glass* it seals the relation list for *The hammer* by closing its left side and unifies *the glass* with the existing structure for *The hammer broke* by donning it a [2] list. (Here is where it gets good) If, however, it realizes that the utterance has ended without a term to complete the mandatory [2] arc of *broke*, the parser can know that it is faced with the unaccusative form of *broke* which has an [2, 1] arc (see Figure 7) and can unify its label with the [1] arc of *The hammer without retracting its previous analysis of hammer or modifying it in any way*.

Unlike LFG, RG is a graph-based theory of grammar that does not attempt to make claims about the neurological implementation of the language faculty. The strata of an RG analysis are generally taken to exist “all at once,” with no attempt made at an account of how they are derived. We see here, though, that a unification-based description of their derivation is *monotonic* and thus *representation-preserving*; it does not have to retract or modify its analyses of sentences containing unaccusative forms of transitive verbs, unlike transformational accounts; it is not difficult to extend the same reasoning to passivization and to more intricate phenomena like dative shift, and the antipassive. This is very appealing characteristic of RG from the perspective of psychological plausibility.

$$\begin{aligned} \textit{Transitive} &:= [H] : P \wedge [1] : T \wedge [2] : T; \\ \textit{Unaccusative} &:= [2, \underline{1}] : T; \end{aligned}$$

Figure 7: SFG rules for transitive and unaccusative

## 4 Relational Grammar and Other Issues in Computational Linguistics

### 4.1 Regularization

In order to be confident enough in our choice of formalism and our performance metrics to assert that we have really characterized language use in what is in some sense the *right* way, we have to be able to prove that our description of the human sentence processor does not depend on language-specific phenomena. Our descriptions must be general enough to account for many languages, since, at birth, it is possible to learn any language, and arguably any second language later on. Relational Grammar provides the computational linguist with analyses of relational phenomena across many languages. Because it does not provide exhaustive accounts of every aspect of natural language syntax, it functions at a higher level of abstraction; it states generalizations in unified rather than disjunctive rules.

Aside from the theoretical elegance of RG analyses, however, taking grammatical roles as first-class primitives is also attractive to the linguistic community for more practical reasons. Researchers working in corpus linguistics and others doing statistical and probabilistic analysis of language face something known as a *sparse data problem*. For example, analyzing the correlation of certain nouns, verbs, and prepositional phrases in an attempt to resolve prepositional phrase attachment ambiguity (Who has binoculars in the sentence *The private eye spied a shady figure with binoculars?*), a researcher stands to benefit from regularization of various alternations of a verb that differ only in their predicate-argument structure. Recall the example of a ditransitive verb that allows dative and passive forms as well; a system that takes explicit notice of grammatical relations during parsing can collapse all of the various forms of the verb and regard them as in some way the same, offsetting the probabilistic unlikelihood of seeing a verb in a dataset, regardless of its form. For an example of how an RG account of roles and relations has been explicitly incorporated into a corpus linguistics project with the goal of regularization, see [9].

### 4.2 The Computational Linguist as Software Engineer: Reuse and Separation of Concerns

Although our primary focus as computational psycholinguists is describing what happens in the brain as humans learn and use language, to do so using a computer is to be a computer programmer. Computer programmers

working in other fields have long understood that writing a large and complex program, and especially a series of related large and complex programs, requires effort and planning. Creating a processing system with broad coverage of a language is certainly a case of writing a large and complex computer program, and there is something about adding to it the capability to process a second, or third, or several more languages is in every sense a process of extending that program.

Software engineering as a collection of techniques for writing computer programs emphasizes the concept of code reuse. Code written for a program should be organized into modules and functions such that a module responsible for performing some abstract task contains only the functions and data it needs in order to do so. Code organized in this way can then be extracted from one program and included in another program that needs to perform the same abstract task. For example, a set of functions written to do linear algebra computations in a program that draws three-dimensional models onto a computer monitor should not have to be rewritten when that programmer or company creates a program to perform bilinear modeling of human color perception. In the same way, when a computational linguist discovers a set of regularities within a language—say, those that apply to auxiliary selection in Italian—those regularities should be incorporated in his processing system in such a way that, when the linguist begins to investigate a new language—say, French—and discovers that the same set of regularities apply, his effort in incorporating them for Italian does not have to be repeated for French.

One way to design systems for reuse is the process of Separation of Concerns (SoC), first described by Edsger Dijkstra in 1974 (published in [5]). SoC is the process or technique of dividing a computer program into features that overlap in functionality or responsibility as little as possible. The goal of SoC is to reduce the complexity of individual divisions. The intuition is that, as computer programs grow into the hundreds of thousands or millions of lines of code, any individual programmer cannot possibly keep the complexity of the entire system ‘in his head’ at one time. While working on one part of the system, he should only have to focus on the relevant aspects of that part. The example of a module that does linear algebra above is an example of SoC. Dividing an application into levels or tiers is another example: a library catalog system should be divided into modules responsible for communicating with patrons and staff, modules that track active loans, etc. Computational linguists should use these same intuitions to guide how they design processing systems.

Treating grammatical roles and surface structure as different and ac-

counting for them differently in a processing system is one way to separate concerns in computational linguistics. A unification parser that has an explicit account of role unification (unify labels) and surface structure unification (tree adjoining and substitution) decouples its description and its implementation of processes and concerns. While it is presently unclear to what degree such decoupling is relevant from a psycholinguistics perspective, it is unarguably attractive from software engineering standpoint.

## 5 Conclusion

Relational Grammar seeks to account for predicate-argument relations across many languages using a small set of primitives with semantic correlates. Its simplicity leaves it lacking in descriptions of constituent structure, but allows for a clear and psychologically plausible account of how PA structure is built during incremental parsing of utterances; it is a *monotonic* model of PA-structure building. It and its philosophy are also attractive from natural language processing and software engineering perspectives.

## References

- [1] Joan Bresnan. *The Mental Representation of Grammatical Relations*. MIT Press series on cognitive theory and mental representation. MIT Press, Cambridge, Mass, 1982.
- [2] Luigi Burzio. *Italian syntax: A Government-Binding approach*. Dordrecht, 1986.
- [3] Vicente Carrillo, , Vicente Carrillo, Vctor J. Daz, and Miguel A. Alonso. A predictive left-corner parser for tree adjoining grammars, 2002.
- [4] William Davies and Carol Rosen. Unions as multi-predicate clauses. *Language*, 64(1):52–88, 1988.
- [5] Edsger Dijkstra. *Selected writings on computing: a personal perspective*. Texts and monographs in computer science. Springer-Verlag, New York, 1982.
- [6] David E. Johnson, Adam Meyers, and Lawrence S. Moss. A unification-based parser for relational grammar. *31st Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pages 97–104, 1993.

- [7] David E. Johnson and Lawrence S. Moss. Some formal properties of stratified feature grammars. *Annals of Mathematics and Artificial Intelligence*, 8:133–173, 1993.
- [8] Mitchell P. Marcus, Donald Hindle, and Margaret M. Fleck. D-theory: Talking about talking about trees. In *21st Annual Meeting of the Association for Computational Linguistics*, pages 129–136, 1983.
- [9] Adam Meyers, Ralph Grishman, and Michiko Kosaka. Formal mechanisms for capturing regularizations. In *Proceedings of the LREC*, 2002.
- [10] Donna Jo Napoli. *The two si’s of Italian: An analysis of reflexive, inchoative, and indefinite subject sentences in Modern Standard Italian*. PhD thesis, Harvard University, 1973.
- [11] David M. Perlmutter. Multiattachment and the unaccusative hypothesis: the perfect auxiliary in italian. *Probus*, 1:63, 1989.
- [12] David M. Perlmutter and Carol G. Rosen, editors. *Studies in Relational Grammar*, volume 2 and 3. University of Chicago Press, 1983.
- [13] Derek Proudian and Carl Pollard. Parsing head-driven phrase structure grammar. In *Proceedings of the 23rd annual meeting on Association for Computational Linguistics*, pages 167–171, Morristown, NJ, USA, 1985. Association for Computational Linguistics.
- [14] Luigi Rizzi. *Issues in Italian Syntax*. Foris, 1982.
- [15] Patrick Sturt and Mathew W. Crocker. Thematic monotonicity. *Journal of Psycholinguistic Research*, 26(3):297–322, 1997.
- [16] Noriko Tomuro. *Left-corner parsing algorithm for unification grammars*. PhD thesis, DePaul University, School of Computer Science, Telecommunications, and Information Systems, 1999.