Chart Parsing and Rule Schemata in PSG

Henry Thompson
Dept. of Artificial Intelligence, Univ. of Edinburgh,
Hope Park Square, Meadow Lane, Edinburgh, EH8 9NW

INTRODUCTION

MCHART is a flexible, modular chart parsing framework I have been developing (in Lisp) at Edinburgh, whose initial design characteristics were largely determined by pedagogical needs.

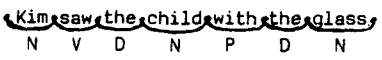
PSG is a grammatical theory developed by Gerald Gazdar at Sussex, in collaboration with others in both the US and Britain, most notably Ivan Sag, Geoff Pullum, and Ewan Klein. It is a notationally rich context free phrase structure grammar, incorporating meta-rules and rule schemata to capture generalisations. (Gazdar 1980a, 1980b, 1981; Gazdar & Sag 1980; Gazdar, Sag, Pullum & Klein to appear)

In this paper I want to describe how I have used MCHART in beginning to construct a parser for grammars expressed in PSG, and how aspects of the chart parsing approach in general and MCHART in particular have made it easy to accommodate two significant aspects of PSG: rule schemata involving variables over categories; and compound category symbols ("slash" categories). To do this I will briefly introduce the basic ideas of chart parsing; describe the salient aspects of MCHART; give an overview of PSG; and finally present the interesting aspects of the parser I am building for PSG using MCHART. Limitations of space, time, and will mean that all of these sections will be brief and sketchy - I hope to produce a much expanded version at a later date.

I. Chart Parsing

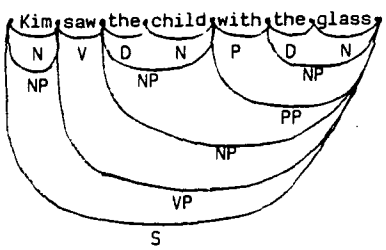
The chart parsing idea was originally conceived of by Martin Kay, and subsequently developed and refined by him and Ron Kaplan (Kay 1973, 1977, 1980; Kaplan 1972, 1973a, 1973b). The basic idea builds on the device known as a well formed substring table, and transforms it from a passive repository of achieved results into an active parsing agent. A well formed substring table can be considered as a directed graph, with each edge representing a node in the analysis of a string. Before any parsing has occurred, all the nodes are (pre)terminal, as in Figure 1.

Figure 1.



Non-terminal nodes discovered in the course of parsing, by whatever method, are recorded in the WFST by the addition of edges to the graph. For example in Figure 2 we see the edges which might have been added in a parsing of the sentence given in Figure 1.

Figure 2.



The advantage of the WFST comes out if we suppose the grammar involved recognises the structural ambiguity of this sentence. If the parsing continued in order to produce the other structure, with the PP attached at the VP level, considerable effort would be saved by the WFST. The subject NP and the PP itself would not need to be reparsed, as they are already in the graph.

What the chart adds to the WFST is the idea of active edges. Where the inactive edges of the WFST (and the chart) represent complete constituents, active edges represent incomplete constituents. Where inactive edges indicate the presence of such and such a constituent, with such and such sub-structure, extending from here to there, active edges indicate a stage in the search for a constituent.

As such they record the category of the constituent under construction, its sub-structure as found so far, and some specification of how it may be extended and/or completed.

The fundamental principle of chart parsing, from which all else follows, is keyed by the meeting of active with inactive edges:

****
The Fundamental Rule
****
Whenever an active edge A and an inactive edge I meet for the first time, if I satisfies A's conditions for extension, then build a* new edge as follows:
- Its left end is the left end of A
- Its right end is the right end of I
- Its category is the category of A
- Its contents are a function (dependent on the grammatical formalism employed) of the contents of A and the category and contents of I
- It is inactive or active depending on whether this extension completes A or not

****

Note that neither A nor I is modified by the above process - a completely new edge is constructed, independent of either of them. In the case of A, this may seem surprising and wasteful of space, but in fact it is crucial to properly dealing with structural ambiguity. It guarantees that all parses will be found, independent of the order in which operations are performed. Whenever further inactive edges are added at this point the continued presence of A, together with the fundamental rule, insures that alternative extensions of A will be pursued as appropriate.

A short example should make the workings of this principle clear. For the sake of simplicity, the grammar I will use in this and subsequent examples is an unadorned set of context free phrase structure rules, and the structures produced are simple constituent structure trees. Nonetheless as should be clear from what follows, the chart is equally useful for a wide range of grammatical formalisms, including phrase structure rules with features and ATNs.

---

*In fact depending on formalism more than one new edge may be built - see below.

Figures 3a-3d show the parsing of "the man" by the rule "NP -> D N". In these figures, inactive edges are light lines below the row of verteces.

Active edges are heavy lines above the row. Figure 3a simply shows the two inactive edges for the string with form-class information.

Figure 3a.

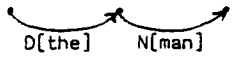
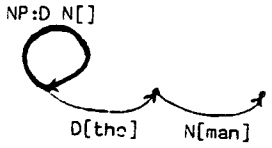


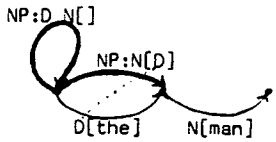
Figure 3b shows the addition of an empty active edge at the left hand end. We will discuss where it comes from in the next section. Its addition to the chart invokes the fundamental rule, with this edge being A and the edge for "the" being I.

Figure 3b.



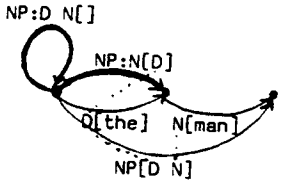
The notation here for the active edges is the category sought, in this case NP, followed by a colon, followed by a list of the categories needed for extension/ completion, in this case D followed by N, followed by a bracketed list of sub-constituents, in this case empty. Since the first symbol of the extension specification of A matches the category of I, an new edge is created by the fundamental rule, as shown in Figure 3c.

Figure 3c.



This edge represents a partially completed NP, still needing an N to complete, with a partial structure. Its addition to the chart invokes the fundamental rule again, this time with it as A and the "man" edge as I. Once again the extension condition is meet, and a new edge is constructed. This one is inactive however, as nothing more is required to complete it.

Figure 3d.



The fundamental rule is invoked for the last time, back at the left hand end, because the empty NP edge (active) now meets the complete NP edge (inactive) for the first time, but nothing comes of this as D does not match NP, and so the process comes to a halt with the chart as shown in Figure 3d.

The question of where the active edges come from is separate from the basic book-keeping of the fundamental principle. Different rule invocation strategies such as top-down, bottom-up, or left corner are reflected in different conditions for the introduction of empty active edges, different conditions for the introduction of empty active edges. For instance for a top-down invocation strategy, the following rule could be used:

****
Top-down Strategy Rule
****
Whenever an active edge is added to the chart, if the first symbol it needs to extend itself is a non-terminal, add an empty active edge at its right hand end for each rule in the grammar which expands the needed symbol.
****

With this rule and the fundamental rule in operation, simply adding empty active edges for all rules expanding the distinguished symbol to the left hand end of the chart will provoke the parse. Successful parses are reflected in inactive edges of the correct category spanning the entire chart, once there is no more activity provoked by one or the other of the rules. Bottom-up invocation is equally straight-forward:

****
Bottom-up Strategy Rule
****
Whenever an inactive edge is added to the chart, for all the rules in the grammar whose expansion begins with the this edge's category, add an empty active edge at its left hand end.
****

Note that while this rule is keyed off inactive edges the top-down rule was triggered by active edges being added. Bottom-up says "Who needs what just got built in order to get started", while top-down says "Who can help build what I need to carry on". Bottom-up is slightly simpler, as no additional action is needed to commence the parse beyond simply constructing the initial chart - the lexically inspired inactive edges themselves get things moving.

Also note that if the grammars to be parsed are left-recursive, then both of these rules need redundancy checks of the form "and no such empty active edge is already in place" added to them.

The question of search strategy is independent of the choice of rule invocation strategy. Whether the parse proceeds depth-first, breadth-first, or in some other manner is determined by the order in which edges are added to the chart, and the order in which active-inactive pairs are considered under the fundamental rule. A single action, such as the adding of an edge to the chart, may provoke multiple operations: a number of edge pairs to be processed by the fundamental rule, and/ or a number of new edges to be added as a result of some rule invocation strategy. Last in first out processing of such multiple operations will give approximately depth-first behaviour, while first in first out will give approximately breadth-first. More complex strategies, including semantically guided search, require more complicated queuing heuristics.

The question of what grammatical formalism is employed is again largely independent of the questions of rule invocation and search strategy. It comes into play in two different ways. When the fundamental rule is invoked, it is the details of the particular grammatical formalism in use which determines the interpretation of the conditions for extension carried in the active edge. The result may be no new edges, if the conditions are not met; one new edge, if they are; or indeed more than one, if the inactive edge allows extension in more than

one way. This might be the case in an ATN style of grammar, where the active edge specifies its conditions for extension by way of reference to a particular state in the network, which may have more than one out-going arc which can be satisfied by the inactive edge concerned. The other point at which grammatical formalism is involved is in rule invocation. Once a strategy is chosen, it still remains each time it is invoked to respond to the resulting queries, e.g. "Who needs what just got built in order to get started", in the case of a simple bottom-up strategy. Such a response clearly depends on the details of the grammatical formalism being employed.

Underlying all this flexibility, and making it possible, is the fundamental rule, which ensures that no matter what formalism, search strategy, and rule invocation strategy* are used, every parse will eventually be found, and found only once.

II. MCHART

In the construction of MCHART, I was principly motivated by a desire to preserve what I see as the principal virtues of the chart parsing approach, namely the simplicity and power of its fundamental principle, and the clear separation it makes between issues of grammatical formalism, search strategy, and rule invocation strategy. This led to a carefully modularised program, whose structure reflects that separation. Where a choice has had to be made between clarity and efficiency, clarity has been preferred. This was done both in recognition of the system's expected role in teaching, and in the hopes that it can be easily adopted as the basis for many diverse investigations, with as few efficiency-motivated hidden biases as possible.

The core of the system is quite small. It defines the data structures for edges and verteces, and organises the construction of the initial chart and the printing of results. Three distinct interfaces are provided which the user manipulates to create the particular parser he wants: A signal table for determining rule invocation strategy; a functional interface for determining grammatical formalism; and a multi-level agenda for determining search strategy.

The core system raises a signal whenever something happens to which a rule invocation strategy might be sensitive, namely the beginning and end of parsing, and the adding of active or inactive edges to the chart. To implement a particular strategy, the user specifies response to some or all of these. For example a bottom-up strategy would respond to the signal Adding InactiveEdge, but ignore the others; while a top-down strategy would need to respond to both AddingActiveEdge and StartParse.

There is also a signal for each new active-inactive pair, to which the user may specify a response. However the system provides a default, which involves the afore-mentioned functional interface. To take advantage of this, the user must define two functions. The first, called ToExtend, when given an active edge and an inactive edge, must return a set of 'rules' which might be used to extend the one over the other. Taken together, an active edge, an inactive edge, and such a rule are called a configuration. The other function the user must define, called RunConfig, takes a configuration as argument and is responsible for implementing the fundamental principle, by building a new edge if the rule applies. For use here and in responses to signals, the system provides the function NewEdge, by which new edges may be handed over for addition to the chart.

The system is embedded within a multi-level agenda mechanism. The adding of edges to the chart, the running of configurations, the raising of signals are all controllable by this mechanism. The user may specify what priority level each such action is to be queued at, and may also specify what ordering regime is to be applied to each queue. LIFO and FIFO are provided as default options by the system. Anything more complicated must be functionally specified by the user.

More detailed specifications would be out of place in this context, but I hope enough has been said to give a good idea of how I have gone about implementing the chart in a clean and modular way. Hardcopy and/or machine-readable versions of the source code and a few illustrative examples of use are available at cost from me to those who are interested. The system is written in ELISP, a local superset of Rutgers Lisp which is very close to Interlisp. A strenuous effort has been made to produce a relatively dialect neutral, transparent implementation, and as the core system is only a few pages long, translation to other versions of Lisp should not be difficult.

III. PSG

Into the vacuum left by the degeneration into self-referential sterility of transformational-generative grammar have sprung a host of non-transformational grammatical theories. PSG, as developed by Gerald Gazdar and colleagues, is one of the most attractive of these. It combines a simplicity and elegance of formal apparatus with a demonstrably broad and arguably insightful coverage of English grammatical phenomena (Gazdar 1980a, 1980b, forthcoming; Gazdar & Sag 1980; Gazdar, Pullum & Sag 1980; Gazdar, Klein, Pullum & Sag forthcoming). It starts with context-free phrase structure rules, with a two bar X-bar category system, under a node admissability interpretation. Four additional notational devices increase the expressive power of the formalism without changing its formal power - features, meta-rules, rule schemata, and compound categories.

The addition of feature marking from a finite set to the category labels gives a large but finite inventory of node labels. Meta-rules are pattern-based rewrite rules which provide for the convenient expression of a class of syntactic regularities e.g. passive and subject-auxilliary inversion. They can be interpreted as inductive clauses in the definition of the grammar, saying effectively "For every rule in the grammar of such and such a form, add another of such and such a form". Provided it does not generate infinite sets of rules, such a device does not change the formal power of the system.

Rule schemata are another notational convenience, which use variables over categories (and features) to express compactly a large (but finite) family of rules. For instance, the rule {S -> NP[PN x] VP[PN x]}** where PN is the person-number feature and x is a variable, is a compact expression of the requirement that subject and verb(-phrase) agree in person-number, and {x -> x and x} might be a simplified rule for handling conjunction.

The final device in the system is a compounding of the category system, designed to capture facts about unbounded dependencies.

This device augments the grammar with a set of derived categories of the form x/y, for all categories x and y in the unaugmented grammar, together with a set of derived rules for expanding these 'slash' categories. Such a category can be interpreted as 'an x with a y

*Defective invocation strategies, which never invoke a needed rule, or invoke it more than once at the same place, can of course vitiate this guarantee.

**Here and subsequently I use old-style category labels as notational equivalents of their X-bar versions.

missing from it'. The expansions for such a category are all the expansions for x, with the '/y' applied to every element on the right hand sides thereof. Thus if {A -> B C} & {A -> D}, then {A/C -> B/C C}, {A/C -> B C/C}, and {A/C -> D/C}. In addition x/x always expands, inter alia, to null. Given this addition to the grammar, we can write rules like {NP -> NP. that S/NP} for relative clauses. If we combine this device with variables over categories, we can write (over-simplified) rules like {S -> x S/x} for topicalization, and {x -> whatever x S/x} for free relatives. This approach to unbounded dependencies combines nicely with the rule schema given above for conjunction to account for the so-called 'across the board' deletion facts. This would claim that e.g. 'the man that Kim saw and Robin gave the book to' is OK because what is conjoined is two S/NPs, while e.g. 'the man that Kim saw and Robin gave the book to Leslie' is not OK because what is conjoined is an S/NP and an S, for which there is no rule.

It is of course impossible to give a satisfactory summary of an entire formalism in such a short space, but I hope a sufficient impression will have been conveyed by the foregoing to make what follows intelligible. The interested reader is referred to the references given above for a full description of PSG by its author(s).

IV. Parsing PSG using MCHART

What with rule schemata and meta-rules, a relatively small amount of linguistic work within the PSG framework can lead to a large number of rules. Mechanical assistance is clearly needed to help the linguist manage his grammar, and to tell him what he's got at any given point. Although I am not convinced there is any theoretical significance to the difference in formal complexity and power between context free grammars and transformational grammars, the methodological significance is clear and uncontestable. Computational tools for manipulating context free grammars are readily available and relatively well understood. On being introduced to PSG, and being impressed by its potential, it therefore seemed to me altogether appropriate to put the resources of computational linguistics at the service of the theoretical linguist. A Parser, and eventually a directed generator, for PSG would be of obvious use to the linguists working within its framework.

Thus my goal in building a parser for PSG is to serve the linguist - to provide a tool which allows the expression and manipulation of the grammar in terms determined by the linguist for linguistic reasons. The goal is not an analogue or "functionally equivalent" system, but one which actually takes the linguists' rules and uses them to parse (and eventually generate).

MCHART has proved to be an exceptionally effective basis for the construction of such a system. Its generality and flexibility have allowed me to implement the basic formal devices of PSG in a non ad-hoc way, which I hope will allow me to meet my goal of providing a system for linguists to use in their day to day work, without requiring them to be wizard programmers first.

Of the four aspects of PSG discussed above, it is rule schemata and slash categories which are of most interest. I intend to handle meta-rules by simply closing the grammar under the meta-rules ahead of time. Feature checking is also straight-forward, and in what follows I will ignore features in the interests of simplicity of exposition.

Let us first consider rule schemata. How are we to deal with a rule with a variable over categories? If we are following a top down rule invocation strategy, serious inefficiencies will result, whether the variable is on the left or right hand sides. A rule with a variable on the left hand side will be invoked by every active edge which needs a non-terminal to extend itself, and a variable on the right hand side of a rule will invoke every rule in the grammar! Fortunately, things are much better under a bottom up strategy. I had already determined to use a bottom up approach, because various characteristics of PSG strongly suggested that, with careful indexing of rules, this would mitigate somewhat the effect of having a very large number of rules.*

Suppose that every rule schema begins with** at least one non-variable element, off which it which it can be indexed.

Then at some point an active edge will be added to the chart, needing a variable category to be extended. If whenever the fundamental rule is applied to this edge and an inactive edge, this variable is instantiated throughout the rule as the category of that inactive edge, then the right thing will happen. The exact locus of implementation is the aforementioned function ToExtend. To implement rule schemata, instead of simply extracting the rule from the active edge and returning it, it must first check to see if the right hand side of the rule begins with a variable. If so, it returns a copy of the rule, with the variable replaced by the category of the inactive edge throughout. In a bottom up context, this approach together with the fundamental rule means that all and only the plausible values for the variable will be tried. The following example should make this clear.

Suppose we have a rule schema for english conjunction as follows: {x -> both x and x}≠, and noun phrase rules including {NP -> Det N}, {NP -> Propn}, {Det -> NP 's}, where we assume that the possessive will get an edge of its own. Then this is a sketch of how "both Kim 's and Robin 's hats" would be parsed as an NP. Figure 4a shows a part of the chart, with the lexical edges, as well as three active edges.

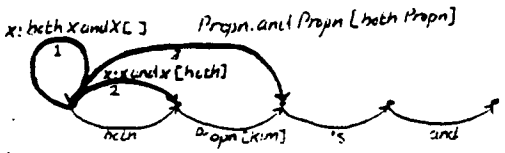


Figure 4a.

*A very high proportion of PSG rules contain at least one (pre)terminal. The chart will support bi-directional processing, so running bottom up all such rules can be indexed off a preterminal, whether it is first on the right hand side or not. For example a rule like {NT -> NT pt NT} could be indexed off pt, first looking leftwards to find the first NT, then rightwards for the other. Preliminary results suggest that this approach will eliminate a great deal of wasted effort.

**In fact given the bi-directional approach, as long as a non-variable is contained anywhere in the rule we are alright. If we assume that the root nature of topicalisation is reflected by the presence in the schema given above of some beginning of sentence marker, then this stipulation is true of all schemata proposed to date.

≠This rule is undoubtedly wrong. I am using it here and subsequently to have a rule which is indexed by its first element. The bi-directional approach obviates the necessity for this, but it would obscure the point I am trying to make to have to explain this in detail.

Edge 1 is completely empty, and was inserted because the conjunction rule was triggered bottom up off the word "both". Edge 2 follows from edge 1 by the fundamental rule. It is the crucial edge for what follows, for the next thing it needs is a variable. Thus when it is added to the chart, and ToExtend is called on it and the Propn edge, the rule returned is effectively {Propn:Propn and Propn [both]}, which is the result of substituting Propn for x throughout the rule in edge 2. This instantiated rule is immediately satisfied, leading to the addition of edge 3. No further progress will occur on this path, however, as edge 3 needs "and" to be extended.

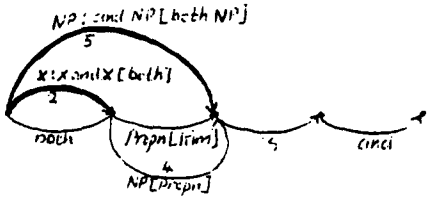


Figure 4b.

Figure 4B shows what happens when at some later point bottom up processing adds edge 4, since a Propn constitutes an NP. Once again the fundamental rule will be invoked, and ToExtend will be called on edge 2 and this new NP edge. The resulting instantiated rule is {NP:NP and NP [both]}, which is immediately satisfied, resulting in edge 5. But this path is also futile, as again an "and" is required.

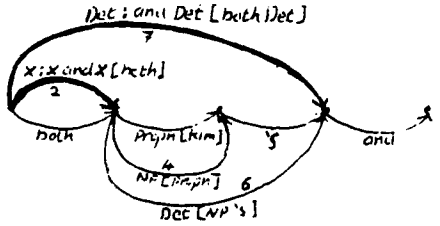


Figure 4c.

Finally Figure 4c shows what happens when further bottom up invocation causes edge 6 to be built - a determiner composed of an NP and a possessive 's. Again the fundamental rule will call ToExtend, this time on edge 2 and this new Det edge. The resulting instantiated rule is {Det:Det and Det [both]}, which is immediately satisfied, resulting in edge 7. From this point it is clear sailing. The "and" will be consumed, and then "Robin 's" as a determiner, with the end result being an inactive edge for a compound determiner spanning "both Kim 's and Robin 's", which will in turn be incorporated into the complete NP.

The way in which the fundamental rule, bottom up invocation, and the generalised ToExtend interact to implement variables over categories is elegant and effective. Very little effort is wasted, in the example edges 3 and 5, but these might in fact be needed if the clause continued in other ways. The particular value of this implementation is that it is not restricted to one particular rule schema. With this facility added, the grammar writer is free to add schemata to his grammar, and the system will accommodate them without any additional effort.

Slash categories are another matter. We could just treat them in the same way we do meta-rules. This would mean augmenting the grammar with all the rules formable under the principles described in the preceding section on PSG. Although this would probably work (there are some issues of ordering with respect to ordinary meta-rules which are not altogether clear to me), it would lead to considerable inefficiency given our bottom up assumption. The parsing of as simple a sentence as "Kim met Robin" would involve the useless invocation of many slash category expanding rules, and a number of useless constituents would actually be found, including two S/NPs, and a VP/NP. What we would like to do is invoke these rules top down. After all, if there is a slash category in a parse, there must be a "linking" rule, such as the relative clause rule mentioned above, which expands a non slash category in terms of inter alia a slash category. Once again we can assume that bottom up processing will eventually invoke this linking rule, and carry it forward until what is needed is the slash category. At this point we simply run top down on the slash category. MCHART allows us to implement this mixed initiative approach quite easily. In addition to responding to the AddingInactiveEdge signal to implement the bottom up rule, we also field the AddingActiveEdge signal and act if and only if what is needed is a slash category. If it is we add active edges for just those rules generated by the slashing process for the particular slash category which is needed. In the particular case where what is needed is x/x for some category x, an empty inactive edge is built as well. For instance in parsing the NP "the song that Kim sang", once the relative clause rule gets to the point of needing an S/NP, various edges will be built, including one expanding S/NP as NP followed by VP/NP. This will consume "Kim" as NP, and then be looking for VP/NP. This will in turn be handled top down, with an edge added looking for VP/NP as V followed by NP/NP among others. "sang" is the V, and NP/NP provokes top down action for the last time, this time simply building an empty inactive edge (aka trace).

The nice thing about this approach is that it is simply additive. We take the system as it was, and without modifying anything already in place, simply add this extra capacity by responding to a previously ignored signal.

Alas things aren't quite that simple. Our implementations of rule schemata and slash categories each work fine independently. Unfortunately they do not combine effectively. NPs like "the song that both Robin wrote and Kim sang" will not be parsed. This is unfortunate indeed, as it was just to account for coordination facts with respect to slashed categories that these devices were incorporated into PSG in the form they have.

The basic problem is that in our implementation of rule schemata, we made crucial use of the fact that everything ran bottom up, while in our implementation of slash categories we introduced some things which ran top down.

The most straight-forward solution to the problem lies in the conditions for the top down invocation of rules expanding slash categories. We need to respond not just to overt slash categories, but also to variables. After all, somebody looking for x might be looking for y/z, and so the slash category mechanism should respond to active edges needing variable categories as well as to those needing explicit slash categories. In that case all possible slash category expanding rules must be invoked. This is not wonderful, but it's not as bad as it might at first appear. Most variables in rule schemata are constrained to range over a limited set of categories. There are also constraints on what slash categories are actually possible. Thus relatively few schemata will actually invoke the full range of slash category rules, and the number of such rules will not be too great either. Although some effort will certainly be wasted, it will still be much less than would have been by the brute force method of simply including the

slash category rules in the grammar directly.

One might hope to use the left context to further constrain the expansion of variables to slash categories, but ordering problems, as well as the fact that the linking rule may be arbitrarily far from the schema, as in e.g. "the song that Kim wrote Leslie arranged Robin conducted and I sang" limit the effectiveness of such an approach.

I trust this little exercise has illustrated well both the benefits and the drawbacks of a mixed initiative invocation strategy. It allows you to tailor the invocation of groups of rules in appropriate ways, but it does not guarantee that the result will not either under-parse, as in this case, or indeed over-parse. The solution in this case is a principled one, stemming as it does from an analysis of the mismatch of assumptions between the bottom up and top down parts of the system.

V. Conclusion

So far I have been encouraged by the ease with which I have been able to implement the various PSG devices within the MCHART framework. Each such device has required a separate implementation, but taken together the result is fully general. Unless the PSG framework itself changes, no further programming is required. The linguist may now freely add, modify or remove rules, meta-rules, and schemata, and the system's behaviour will faithfully reflect these changes without further ado. And if details of the framework do change, the effort involved to track them will be manageable, owing to the modularity of the MCHART implementation. I feel strongly that the use of a flexible and general base such as MCHART for the system, as opposed to custom building a PSG parser from scratch, has been very much worth while. The fact that the resulting system wears its structure on its sleeve, as it were, is easily explained and (I hope) understood, and easily adapted, more than offsets the possible loss of efficiency involved.

The reinvention of the wheel is a sin whose denunciations in this field are exceeded in number only by its instances. I am certainly no less guilty than most in this regard. None the less I venture to hope that for many aspects of parsing, a certain amount of the work simply need not be redone any more. The basic conceptual framework of the chart parsing approach seems to me ideally suited as the starting point for much of the discussion that goes on in the field. A wider recognition of this, and the wide adoption of, if not a particular program such as MCHART, which is too much to expect, then at least of the basic chart parsing approach, would improve communications in the field tremendously, if nothing else. The direct comparison of results, the effective evaluation of claims about efficiency, degrees of (near) determinism, etc. would be so much easier. The chart also provides to my mind a very useful tool in teaching, allowing as it does the exemplification of so many of the crucial issues within the same framework.

Try it, you might like it.

In the same polemical vein, I would also encourage more cooperation on projects of this sort between theoretical and computational linguists. Our technology can be of considerable assistance in the enterprise of grammar development and evaluation. There are plenty of other non-transformational frameworks besides PSG which could use support similar to that which I am trying to provide. The benefit is not just to the linguist - with a little luck in a few years I should have the broadest coverage parser the world has yet seen, because all these linguists will have been using my system to extend their grammar. Whether I will actually be able to make any use of the result is admittedly less than clear, but after all, getting there is half the fun.

VI. References

Gazdar, G.J.M. (1980a) A cross-categorial semantics for coordination. Linguistics & Philosophy 3, 407-409.

_____ (1980b) Unbounded dependencies and coordinate structure. To appear in Linguistic Inquiry 11.

_____ (1981) Phrase Structure Grammar. To appear in P. Jacobson and G.K. Pullum (eds.) The nature of syntactic representation.

_____, G.K. Pullum, & I. Sag (1980) A Phrase Structure Grammar of the English Auxiliary System. To appear in F. Heny (ed.) Proceedings of the Fourth Groningen Round Table.

_____, G.K. Pullum, I. Sag, & E.H. Klein (to appear) English Grammar.

Kaplan, R.M. (1972) Augmented transition networks as psychological models of sentence comprehension. Artificial Intelligence 3, 77-100.

_____ (1973a) A General Syntactic Processor. In Rustin (ed.) Natural Language Processing. Algorithmics Press, N.Y.

_____ (1973b) A multi-processing approach to natural language. In Proceedings of the first National Computer Conference. AFIPS Press, Montvale, N.J.

Kay, M. (1973) The MIND System. In Rustin (ed.) Natural Language Processing. Algorithmics Press, N.Y.

_____ (1977) Morphological and syntactic analysis. In A. Zampolli (ed.) Syntactic Structures Processing. North Holland.

_____ (1980) Algorithm Schemata and Data Structures in Syntactic Processing. To appear in the proceedings of the Nobel Symposium on Text Processing 1980. Also CSL-80-12, Xerox PARC, Palo Alto, CA.