Chart Parsing Lambek Grammars: Modal Extensions and Incrementality

Mark Hepple
Cambridge University Computer Laboratory, Cambridge, UK.

Abstract

This paper[1] describes a method for chart parsing Lambek grammars. The method is of particular interest in two regards. Firstly, it allows efficient processing of grammars which use necessity operators, an extension proposed for handling locality phenomena. Secondly, the method is easily adapted to allow incremental processing of Lambek grammars, a possibility that has hitherto been unavailable.

Introduction

Categorial Grammars (CGs) consist of two components: (i) a lexicon, which assigns syntactic types (plus an associated meaning) to words, (ii) a calculus which determines the set of admitted type combinations. The set of types (T) is defined recursively in terms of a set of basic types (T_0) and a set of operators ({\\,/} for standard *bidirectional* CG), as the smallest set such that (i) $T_0 \subseteq T$, (ii) *if* x,y $\in$ T, *then* x\\y, x/y $\in$ T.[2] Intuitively, lexical types specify subcategorisation requirements of words, and requirements on constituent order. We here address a particular *flexible* CG, the (product-free) Lambek calculus (**L**: Lambek, 1958). The rules below provide a *natural deduction* formulation of **L** (Morrill et al. 1990; Barry et al. 1991), where dots above a type represent a proof of that type. Proofs proceed from a number of initial assumptions, consisting of individual types, some of which may be "discharged" as the proof is constructed. Each type in a proof is associated with a lambda expression, corresponding to its meaning. The elimination rule /E states that proofs of A/B and B may be combined to construct a proof of A. The introduction rule /I indicates that we may discharge an assumption B within a proof of A to construct a proof of A/B (square brackets indicating the assumption's discharge). There is a side condition on the introduction rules, reflecting the ordering significance of the directional slashes. For /I (resp. \\I), the assumption discharged must be the rightmost (resp. leftmost) undischarged assumption in the proof. Elimination and introduction inferences correspond semantically to steps of functional application and abstraction, respectively.

Hypothesis rule: A:x

Elimination rules:

$$\frac{\begin{matrix}\vdots\\ \mathrm{A/B}{:}f\end{matrix}\quad\begin{matrix}\vdots\\ \mathrm{B}{:}x\end{matrix}}{\mathrm{A}{:}fx}/\mathrm{E}\qquad\qquad \frac{\begin{matrix}\vdots\\ \mathrm{B}{:}x\end{matrix}\quad\begin{matrix}\vdots\\ \mathrm{A\backslash B}{:}f\end{matrix}}{\mathrm{A}{:}fx}\backslash\mathrm{E}$$

Introduction rules:

$$\frac{\begin{matrix}\ldots[\mathrm{B}{:}x]^{i}\\ \vdots\\ \mathrm{A}{:}f\end{matrix}}{\mathrm{A/B}{:}\lambda x.f}/\mathrm{I}^{i}\qquad\qquad \frac{\begin{matrix}[\mathrm{B}{:}x]^{i}\ldots\\ \vdots\\ \mathrm{A}{:}f\end{matrix}}{\mathrm{A\backslash B}{:}\lambda x.f}\backslash\mathrm{I}^{i}$$

Each proof demonstrates the possibility of combining the types of its undischarged assumptions, in their given order, to yield the type at the bottom of the proof. The following proof of "simple forward composition" illustrates the approach.

$$\cfrac{\cfrac{\mathrm{a/b}{:}x\quad \cfrac{\mathrm{b/c}{:}y\quad [\mathrm{c}{:}z]^{i}}{\mathrm{b}{:}(y\,z)}/\mathrm{E}}{\mathrm{a}{:}(x\,(y\,z))}/\mathrm{E}}{\mathrm{a/c}{:}\lambda z.(x\,(y\,z))}/\mathrm{I}^{i}$$

Following Prawitz (1965), a *normal form* (NF) for proofs can be defined using the following meaning preserving contraction rule and its mirror image dual with \\ in place of /, which, under a version of the Curry-Howard correspondence between proofs and lambda terms, are analogous to the β-contraction rule $((\lambda x.\mathrm{P})\mathrm{Q} \rhd \mathrm{P}[\mathrm{Q}/x])$ for lambda expressions.

$$\cfrac{\cfrac{\begin{matrix}\ldots[\mathrm{B}]^{i}\\ \vdots\\ \mathrm{A}\end{matrix}}{\mathrm{A/B}}/\mathrm{I}^{i}\quad \begin{matrix}\vdots\\ \mathrm{B}\end{matrix}}{\mathrm{A}}/\mathrm{E}\qquad \rhd \qquad \begin{matrix}\vdots\\ \ldots\mathrm{B}\\ \vdots\\ \mathrm{A}\end{matrix}$$

Under this system, every **L** proof has an equivalent 'β-NF' proof. Such β-NF proofs have a straightforward structural characterisation, that their *main branch* (the unique path from the proof's end-type to an assumption, that includes no types forming the argument for an elimination step) consists of a sequence of (≥ 0) eliminations followed by a sequence of (≥ 0) introductions.

[1] I am grateful to Esther König for discussion of the paper. The work was done under a grant to the Cambridge University Computer Laboratory, "Unification-based models of lexical access and incremental interpretation", SPG 893168.

[2] In this notation, x/y and x\\y are both functions from y into x. A convention of *left association* is used, so that, e.g. ((s\\np)/pp)/np may be written s\\np/pp/np.

The main approach for parsing L has been *sequent calculus* theorem proving.[3] Used naively, this approach is inefficient due to 'spurious ambiguity', i.e. the existence of multiple equivalent proofs for combinations. König (1989) and Hepple (1990a) develop a solution to this problem based on defining a NF for sequent proofs. These NF systems as yet cover only the basic calculus, and do not extend to various additions proposed to overcome the basic system's shortcomings as a grammatical framework.

Some importance has been attached to the properties of flexible CGs in respect of *incremental* processing. These grammars typically allow sentences to be given analyses which are either fully or primarily left-branching, in which many sentence-initial substrings are interpretable constituents, providing for processing in which the interpretation of a sentence is generated 'on-line' as the sentence is presented. Incrementality is characteristic of human sentence processing, and might also allow more efficient machine processing of language, by allowing early filtering of semantically implausible analyses. It is notable, however, that *no* methods have yet been proposed for incremental parsing of Lambek grammars. In what follows, I describe a chart method for L and then show how it may be modified to allow both inclusion of an operator □, used for handling locality constraints, and also to allow incremental parsing of L.

Chart Parsing Lambek Grammars

Standard chart methods are inadequate for L because proving that some combination of types is possible may involve 'hypothetical reasoning', i.e. using additional assumptions over and above just the types that are being combined. For example, the above proof of a/b, b/c ⇒ a/c requires an additional assumption c, subsequently discharged. Standard chart parsing involves ordering the edges for lexical categories along a single dimension, and then adding edges for constituents that span wider substretches of this dimension as constituents are combined. The problem for L is that there is no place in this set up for additional hypothetical elements. Placing edges for them anywhere on the single dimension of a normal chart would simply be incorrect.

König (1990, 1991), in the only previous chart method for L, handles this problem by placing hypothetical elements on separate, independently ordered 'minicharts', which are created ('emitted') in response to the presence of edges that bear 'higher order' functor types (i.e. seeking arguments having functional types), which may require 'hypothetical reasoning' in the derivation of their argument. Minicharts may 'attach' themselves into other charts (including other minicharts) at points where combinations are possible, so that 'chains of attachment' may arise. Some fairly complicated book-keeping is required to keep track of what has combined with what as a basis for ensuring correct 'discharge' of hypothetical elements. This information is encoded into edges by replacing the simple indices (or vertices) of standard charts with 'complex indices'. Unfortunately, the complexity of this method precludes a proper exposition here. However, some differences between König's method and the method to be proposed will be mentioned at the end of the next section.

[3] Space limits preclude discussion of recent proof net work.

A New Chart Approach

I next present a new chart parsing method for L. Its most striking difference to the standard approach is that there is typically more than one ordering governing the association of edges in a chart. These orderings intersect and overlap, making a chart a 'multidimensional object'. A second difference is that the basic unit we adopt for specifying the orderings of the chart is *primitive intervals*, rather than point-like vertices, where the relative order of the primitive intervals that make up an ordering must be explicitly defined. The span of edges is specified extensionally as the concatenated sum of some number of primitive intervals. The method is perhaps most easily explained by example.

To parse the combination x/y, y/z, z ⇒ x, we require a three element ordering `ordering(a.b.c)` (a b and c being primitive intervals). The three types give three edges, each having three fields: (i) the edge's span (here a primitive interval), (ii) its type (iii) the type's 'meaning' (here a unique constant).

```
[a, x/y, t1]
[b, y/z, t2]
[c, z, t3]
```

Edges are combined under the following chart rules, corresponding to our elimination rules:

```
if      [i, X/Y, A]  and  [j, Y, B]
        and isa_subord(i.j)
then    [i.j, X, (AB)]

if      [i, Y, B]  and  [j, X\Y, A]
        and isa_subord(i.j)
then    [i.j, X, (AB)]
```

The rules allow two edges with appropriate types to combine provided that the concatenation of their spans is a substring of some defined ordering (a test made by the predicate `isa_subord`). Given these rules, our chart will expand to include the following two edges. The presence of an edge with type x that spans the full width of the single defined ordering shows that x/y, y/z, z ⇒ x can be derived.

```
[b.c, y, (t2 t3)]
[a.b.c, x, (t1 (t2 t3))]
```

```
emit([H,T,_]):
  if   T = X/(Y\B1/F1....\Bn/Fm),
  then (add_edges:  [i1,B1,v1],  ...., [in,Bn,vn], [jm,Fm,wm],  ...., [ji,F1,w1]
        add_condition:
           if ordering(P.H.Q.R) and non_empty(Q)  then ordering(i1...in.Q.jm...j1)
        add_condition:   if [(i1...in.K.jm...j1),Y,S]  and isa_subord(H.K)
                         then [K, (Y\B1/F1....\Bn/Fm), (wm@vn@....w1@v1@S)]])
  else
  if   T = X\(Y\B1/F1....\Bn/Fm),
  then (add_edges:  [i1,B1,v1],  ...., [in,Bn,vn], [jm,Fm,wm],  ...., [ji,F1,w1]
        add_condition:
          if ordering(P.Q.H.R) and non_empty(Q) then ordering(i1...in.Q.jm...j1)
        add_condition:   if [(i1...in.K.jm...j1),Y,S] and isa_subord(H.K)
                         then [K, (Y\B1/F1....\Bn/Fm), (wm@vn@....w1@v1@S)]])
```

Figure 1: The EMIT procedure

Our next example x/(y\p/q), y/z\p, z/q ⇒ x requires 'hypothetical reasoning' in its derivation, which is made possible by the presence of the higher-order functor x/(y\p/q). In the natural deduction approach, deriving the functor's argument y\p/q might involve introduction inference steps which discharge additional assumptions p and q occurring peripherally within the relevant subproof. To chart parse the same example, we require firstly the following three edges and require a three element ordering:

```
ordering(a.b.c)
[a, x/(y\p/q), t1]
[b, y/z\p, t2]
[c, z/q, t3]
```

As in König's approach, an 'emit' step is performed on the edge which bears a higher-order type, giving various additions to the chart needed to allow for hypothetical reasoning. Firstly, this gives two new edges, which are assigned new primitive intervals:

```
[d, p, v1]
[e, q, v2]
```

Some new orderings must be defined to allow these edges to combine. Since the higher-order functor is forward directional, possible arguments for it must occupy non-empty intervals H such that isa_subord(a.H). Hypothetical reasoning with the two new edges is useful only in so far as it contributes to deriving edges that occupy these spans. Hence, the required new orderings are (d.H.e) such that isa_subord(a.H). Such new orderings are most conveniently created by including the following *condition on orderings*.

```
if    ordering(P.a.Q.R) and non_empty(Q)
then  ordering(d.Q.e)
```

In general, such conditions may fire after the emit step that creates the condition, when other new orderings are created that include the emitting edge's span. The above condition causes new orderings (d.b.e) and (d.b.c.e) to be defined, allowing combinations that yield the following edges:

```
[d.b, y/z, (t2 v1)]
[c.e, z, (t3 v2)]
[d.b.c.e, y, ((t2 v1)(t3 v2))]
```

The final thing created by the emit process is the following *condition on edges* (where A@B represents lambda abstraction over A in B):

```
if    [d.D.e, y, S] and isa_subord(a.D)
then  [D, y\p/q, v2@v1@S]
```

This condition has the effect that whenever an edge of a certain form is added to the chart, another edge of a related form is also added. The condition completes the process of hypothetical reasoning, by syntactically and semantically abstracting over the hypothetical elements ('discharging' them) to derive the function required as argument by the higher order functor. Note that, since the intervals d and e are unique to the two edges created in the emit process, any edge spanning an interval (d.D.e) must have involved these two edges in its derivation. The condition 'fires' on the above edge spanning (d.b.c.e) to give the first of the following edges, which by combination gives the second. This final edge demonstrates the derivability of original goal combination.

```
[b.c, y\p/q, v2@v1@((t2 v1)(t3 v2))].
[a.b.c, x, (t1 v2@v1@((t2 v1)(t3 v2)))].
```

We have now seen all the basic ingredients required for handling hypothetical reasoning in the new approach. Figure 1. shows a general (if still somewhat informal) statement of the emit procedure which is called on every edge added to the chart, but which only has an effect when an edge bears a higher-order type. The specific consequences depend on the functor's directionality. The notation (Y\B1/F1....\Bn/Fm) stands for a functional type requiring n backward directional arguments B1....Bn and m forward directional arguments F1....Fm in any order.[4] In each case, the procedure simply adds an edge for each required hypothetical element, a condition on orders (to create all required new orderings), and a condition on edges, which fires to produce an edge for the result of hypothetical reasoning, should it succeed. Note that edges produced by such conditions are there only to be argument to some higher order functor, and allowing them combine with other edges *as functor* would be unnecessary work. I assume that such edges are marked, and that some mechanism operates to block such combinations.[5]

A slightly modified emit procedure is required to allow for deriving overall combinations that have a functional result type. I will not give a full statement of this procedure, but merely illustrate it. For example, in proving a combination $\Gamma \Rightarrow$ y\p/q, where an ordering Q had been defined for the edges of the types Γ, emitting the result type y\p/q would give only a single new ordering (not a *condition* on orderings), a condition on edges, and two new edges for the hypothetical elements as follows:

```
ordering(a.Q.b)

if   [a.Q.b, y, S]
then [Q, y\p/q, v2@(v1@S)]

[a, p, v1]
[b, q, v2]
```

That completes description of the new chart method for L. A few final comments. Although the method has been described for proving type combinations, it can also be used for parsing word strings, since lexical ambiguity presents no problems. Note that defining a new ordering may enable certain combinations of edges already present in the chart that were not previously allowed. However, simply checking for all edge combinations that the new ordering allows will result in many previous combinations being redone, since new orderings always share some suborderings with previously defined orderings. One way to avoid this problem is to only check for combinations allowed by substrings of the new ordering that were not previously suborderings.

Concerning the soundness of this method, note that chart derivations can be easily translated into (correct) natural deduction proofs, given a knowledge of which edges gave rise to which others, i.e. with binary edge combinations corresponding to elimination inferences, and with the creation of an edge by a condition on edges corresponding to some sequence of introduction inferences. In fact, chart derivations all translate to β-NF proofs, i.e. with introductions always made after any eliminations on the main branch of any subproof. This observation provides at least an informal indication of the completeness of the method, since the mechanisms described should allow for chart derivations corresponding to *all* possible β-NF proofs of a given combination, which (as we noted earlier) are fully representative.

Another issue is whether the method is *minimal* in the sense of allowing only a single chart derivation for each reading of a combination. This is not so, given that distinct but equivalent β-NF proofs of a combination are possible, due to a second source of equivalence for proofs analogous to η-equivalence of lambda expressions (i.e. that $f = \lambda x.fx$). For example, the combination a/(b/c), b/c $\Rightarrow$ a has two β-NF proofs, one involving 'unnecessary' hypothetical reasoning. However, having equivalent edges represented on the chart, and the undesirable consequences for subsequence derivations, can be avoided by a simple identity check on edge addition, provided that the meaning terms of edges produced by conditions on edges are subject to η-normalisation.

I will finish with some comparisons of the method to that of könig (1990, 1991). The importance of König's method as precursor for the new method cannot be overstated. However, the new approach is, I believe, conceptually much simpler than König's. This is largely due to the use of 'conditions on edges' in the new approach to handle discharge of hypothetical elements, which allows edges to be much simpler objects than in König's approach, where edges instead have to encode the potentially complex information required to allow proper discharge in their 'complex indices'. The complex nature of König's edges considerably obscures the nature of parsing as being simply reasoning about sequences of types, and also makes it difficult to see how the method might be adapted to allow for extensions of L involving additional operators, even ones that have straightforward sequent rules.

A second difference of the new method is that orderings that govern the association of edges are explicitly defined. There is a sense in which the multiple intersecting orderings of the new approach can be seen to express the dimensions of the search space addressed in sequent calculus theorem proving, although collapsing together the parts of that search space that have common structure. In König's method, although the elements that belong together in a minichart are relatively ordered, the attachment of one minichart to another is allowed wherever relevant edges can combine (although subject to some constraints preventing infinite looping). This means that elements may be

[4]This notation is rather clumsy in that it appears to suggest the presence of at least one forward and one backward directional argument and also a relative ordering of these arguments, when neither of these implications is intended. A similar point can be made about abstractions in the schematic semantics vm@vn@....v1@v1@S, whose order and number will in fact mirror that of the corresponding syntactic arguments. A more satisfactory statement of the emit procedure could be made recursively, but this would take up too much space.

[5]An alternative would be not entering such edges at all, but instead have a condition on edges that creates an edge for the result of combining the emitting higher-order functor with its implicitly derived argument, directly.

combined that would not be in sequent calculus theorem proving or in the new chart method. The consequences of this difference for the relative complexity of the two chart methods is at present unknown.

Parsing Modal Extensions

Various extensions of the basic Lambek calculus have been proposed to overcome some of its limitations as a grammatical approach. Morrill (1989, 1990) suggests a unary operator $\Box$, for handling locality constraints on binding and reflexivisation. This has the following inference rules, which make it behave somewhat like necessity in the modal logic S4:

$$\frac{\begin{matrix}\vdots\\ \Box A\end{matrix}}{A}\,\Box E \qquad \frac{\begin{matrix}\vdots\\ A\end{matrix}}{\Box A}\,\Box I$$

where every undischarged assumption is a $\Box$-type

I will try to briefly suggest how $\Box$ may help in handling locality constraints. Boundaries arise where lexical functors seek a modal argument, i.e. are of the form $x/\Box y$, the presence of the $\Box$ making the argument phrase potentially a bounded domain. In addition, all lexical types are of the form $\Box x$, i.e. have a single $\Box$ as their outermost operator, which allows them to appear embedded within modal domains (c.f. the $\Box$I rule's requirement of $\Box$-ed assumptions). For example, a lexical NP might be $\Box$np, a transitive verb $\Box$(s\np/np), and a sentence-complement verb like *believes* type $\Box$(s\np/$\Box$s). In a standard flexible CG treatment, extraction is handled by functional abstraction over the position of the missing (i.e. extracted) element. The type of the abstracted element is determined by the type of the higher order lexical type that requires this abstraction, e.g. the relative pronoun type rel/(s/np) abstracts over np. Note that this relative pronoun type cannot extract out of an embedded modal domain, because it abstracts over a bare (i.e. non-modal) np, whose presence would block $\Box$I rule's use in deriving the overall modal constituent. However, a relative pronoun rel/(s/$\Box$np), which abstracts over a modal type $\Box$np, *can* extract out of an embedded modal domain.

Including this operator presents considerable problems for efficient processing. Firstly, it excludes the use of the NF systems devised for the calculus (König, 1989; Hepple, 1990a). As noted above, spurious ambiguity makes ordinary (i.e. non-normal form) sequent theorem proving of L inefficient. This problem is greatly increased by inclusion of $\Box$, largely due to non-determinism for use of the $\Box$E rule.[6]

[6]Consider a sequent $S = \Box x_1, \Box x_2, \ldots, \Box x_n \Rightarrow x_0$, where the related sequent $S' = x_1, x_2, \ldots, x_n \Rightarrow x_0$ is a theorem. Non-determinism for use of [$\Box$L] means that there are $n!$ different paths of inference from S to S′, so that there are at least $n!$ proofs of S for each proof of S′. In fact, interaction of [$\Box$L] with other inference rules means that there is typically many more proofs than this.

The new chart method is fairly easily adapted to allow for $\Box$, avoiding the non-determinism problem of the sequent system, so that parsing examples with $\Box$ is typically only slightly slower than parsing related examples without any $\Box$s. Firstly, it is crucial that we can always identify the parent edge(s) for some edge (i.e. the immediate edge(s) from which it is derived), and thereby an edge's more distant ancestors. I ignore here the precise details of how this is done. The following chart rule serves in place of the $\Box$E rule:

```
if   [i, □X, A] then [i, X, A]
```

For the combination $x/\Box y$, $\Box(y/z)$, $\Box z \Rightarrow x$, we would require the following ordering and first three edges. The next three edges then result from the operation of chart rules:

```
ordering(a.b.c)
[a, x/□y, t1]
[b, □(y/z), t2]
[c, □z, t3]
[b, y/z, t2]
[c, z, t3]
[(b.c), y, (t2 t3)]
```

To allow completion, we must extend the emit procedure to also take action in cases where the type of an added edge seeks a modal argument. For the case at hand, the emit procedure would create a condition on edges as follows:

```
if   [H, y, S] and isa_subord(a.H)
and  check_modal_history([H, y, S])
then [H, □y, S]
```

The procedure `check_modal_history` used by this condition checks the edge's 'history' to see if it has appropriate ancestors to license the $\Box$-introduction step. Recall that the $\Box$I rule requires that the undischarged assumptions of the proof to which it applies are all $\Box$-types. The corresponding requirement for the chart system is that the edge must have ancestors with $\Box$-types that together span the full width of the edge's span `H` (i.e. there must be a subset of the edge's ancestor edges that have $\Box$-types, and whose spans concatenate to give `H`). The edge `[(b.c), y, (t2 t3)]` satisfies this requirement, and so the condition will fire, allowing the parse to proceed to successful completion, as follows:

```
[(b.c), □y, (t2 t3)]
[(a.b.c), x, (t1 (t2 t3))]
```

More complicated cases arise when an emitted functor seeks an argument type that is both functional and modal. As suggested above, a satisfactory statement of the emit process is best given recursively, but there is not sufficient space here. Hopefully, an example will adequately illustrate the method. Consider what is required in emitting an edge `[a, w/(□(x\y)/z), t1]`, whose type seeks an argument ($\Box$(x\y)/z, i.e. a function to a modal form of a further functional type. As before, emitting creates two new edges and a single condition on orderings:

```
[i, y, v1]
[j, z, v2]

if    ordering(P.a.Q.R) and non_empty(Q)
then  ordering(i.Q.j)
```

However, recursive decomposition of the type $((\Box(x\backslash y)/z)$ gives rise to *three* separate conditions on edges (which reflect the three aspects of the description of this type as a 'function to a modal form of a further functional type'):

```
if   [(i.H.j), x, S] and isa_subord(a.H)
then [(H.j), x\y, v1@S]
if   [H.j, x\y, S] and isa_subord(a.H)
and  check_modal_history([H.j, x\y, S])
then [H.j, [](x\y), S]
if   [H.j, [](x\y), S] and isa_subord(a.H)
then [H, [](x\y)/z, v2@S]
```

These three conditions 'chain' together to create edges with the type required by the emitted functor. Of course in practice, the three conditions could be collapsed into a single condition, and such a move seems sensible from the viewpoint of efficiency.

Incremental Parsing

Despite considerable interest in the theoretical possibility of incremental processing using the Lambek calculus, no incremental parsing methods have as yet been proposed. Indeed, most Lambek parsing work has been based around sequent theorem proving, which might be viewed as antithetical to incremental processing since it fundamentally involves reasoning about complete sequences of types. In fact it is fairly easy to modify the chart method to allow some extent of incremental processing, i.e. so that scanning left-to-right through an input string (or type sequence), the chart will contain edges assigning types to substrings that would not otherwise receive types during parsing, including some for initial substrings of the input.

The modification of the chart method involves allowing an additional extent of hypothetical reasoning over that so far allowed, so that edges for hypothetical types are added not only for higher-order functors, but also for first-order functors. This is allowe by a new procedure **emit***, described below. **emit*** is called on every edge added to the chart, but only has an effect if the edge's type is functional, creating a new edge for a hypothetical type corresponding to the function's first argument, as well as a condition on orderings and one on edges. The condition on orderings creates new orderings allowing the hypothetical edge to combine with its 'emittor', and the result of that combination to be combined with further edges. (The requirement `J \= i.K` prevents the condition incorrectly reapplying to its own output.) Note that the new edge's interval is peripheral in the new orderings that are defined since it is only in peripheral position that the new hypothesis can be discharged (hence, we have `(G.H.i)` in the condition of the first case rather than `(G.H.i.J)`). Such discharge is made by the new condition on edges.

```
emit*([H,T,_]):
  if    T = X/Y
  then  add_edge:  [i, Y, v]
        add_condition:
          if    ordering(G.H.J) and  J \= i.K
          then ordering(G.H.i)
        add_condition:
          if    [(Q.i),Z,S] and  non_empty(Q)
          then [Q,Z/Y,v@S]
  else
   if    T = X\Y
   then  add_edge:  [i, Y, v]
         add_condition:
           if    ordering(G.H.J) and  G \= K.i
           then ordering(i.H.J)
         add_condition:
           if    [(i.Q),Z,S] and  non_empty(Q)
           then [Q,Z\Y,v@S]
```

Let us look at some examples (where we limit our attention to just edges relevant to the discussion). Consider parsing the type sequence (x/y, y/z, z). Since the method should not depend on the parser knowing the length of the input sequence in advance, an ordering will be defined with each scanning step that covers just the material so far scanned, and which extends the ordering of the previous scanning step by one. After scanning the first two types of the input, the chart will include at least the following two edges and ordering:

```
ordering(a.b)
[a, x/y, t1]
[b, y/z, t2]
```

Applying **emit*** to the second edge (ignoring the first edge here) yields the following edge and conditions:

```
[i, z, v]
if    ordering(G.b.J) and  J \= i.K
then  ordering(G.b.i)
if    [(Q.i),T,S] and  non_empty(Q)
then  [Q,T/z,v@S]
```

The condition on orderings will fire on the ordering `(a.b)` to produce a new ordering `(a.b.i)`, which permits the first two of the following edges to be built, the third being generated from the second by the condition on edges. The type x/z this edge assigns to the initial substring (x/y, y/z) of the input (corresponding to the composition of the two functions) would not have been created during parsing with other Lambek parsing methods.

```
[(b.i), y, (t2 v)]
[(a.b.i), x, (t1 (t2 v))]
[(a.b), x/z, v@(t1 (t2 v))]
```

As a second example, consider the stage of having scanned the first two types of the input sequence (y, x\y/z, z). Scanning yields the following ordering and the first two edges. Applying **emit*** to the second edge yields the third edge, and two conditions:

```
ordering(a.b)
[a, x/y, t1]
[b, y/z, t2]
[i, z, v]
if    ordering(G.b.J) and  J \= i.K
then  ordering(G.b.i)
if    [(Q.i),T,S] and non_empty(Q)
then  [Q,T/z,v@S]
```

The ordering condition gives the following new ordering, allowing creation of the subsequent new edges. As before, the last edge assigns a type to the combination of the first two input types which would not otherwise be expected during parsing.

```
ordering(a.b.i)
[(b.i), x\y, (t2 v)]
[(a.b.i), x, ((t2 v) t1)]
[(a.b), x/z, v@((t2 v) t1)]
```

Although the method allows for a considerable degree of incrementality, some conceivable incremental constituents will not be created that would be in parsing with alternative categorial frameworks. For example, rules of type raising and composition in *Combinatory Categorial Grammar* (Steedman, 1987; Szabolcsi, 1987) would allow incremental combination of types vp/s, np $\Rightarrow$ vp/(s\np), not allowed by the present approach. The modified chart method instead allows for the construction of incremental constituents in a manner that most closely relates to the notion of *dependency constituency* argued for by Barry & Pickering (1990) (see also Hepple, 1991), although since the modified parser is still a complete parser for **L** it *cannot* be viewed as implementing a notion of dependency constituency.[7] Finally, it should be noted that the additional hypothetical reasoning allowed by **emit*** and combinations involving additional 'incremental constituents' result in many 'spurious' analyses, so that the incremental chart method is in general slower than the non-incremental chart method.

Conclusion

I have presented a chart parsing method for the Lambek calculus, which I would argue has several advantages over that of König (1990, 1991). Firstly, I believe that it is considerably conceptually clearer than König's method, and more straightforwardly reflects intuitions about the nature of hypothetical reasoning in proving **L** combinations. Secondly, the relatively straightforward nature of the system with respect to reasoning about sequences of types should, I believe, make it easier to adapt the method to allow for additional type-forming operators over those already provided in the (product-free) Lambek calculus, particularly where operators have fairly straightforward sequent rules. We have seen how the method can be extended to allow for Morrill's $\Box$operator. We have also seen how the method may be modified to allow incremental parsing of Lambek grammars.

[7]However, some version of a chart parser that used only the kind of hypothetical reasoning allowed by the **emit*** procedure, and not that of the **emit** procedure, might well implement a notion of dependency constituency.

References

Barry, G., Hepple, M., Leslie, N. and Morrill, G. 1991. 'Proof Figures and Structural Operators for Categorial Grammar', *Proc. of EACL-5.*.

Barry, G. and Morrill, G. 1990. (Eds). *Studies in Categorial Grammar.* Edinburgh Working Papers in Cognitive Science, Volume 5. Centre for Cognitive Science, University of Edinburgh.

Barry, G. and Pickering, M. 1990. 'Dependency and Constituency in Categorial Grammar', in Barry and Morrill, 1990.

Hepple, M. 1990a. 'Normal form theorem proving for the Lambek calculus', *Proc. of COLING-90.*

Hepple, M. 1990b. *The Grammar and Processing of Order and Dependency: A Categorial Approach.* Ph.D. dissertation, Centre for Cognitive Science, University of Edinburgh.

Hepple, M. 1991. 'Efficient Incremental Processing with Categorial Grammar', *Proc. of ACL-27.*

König, E. 1989, 'Parsing as natural deduction', *Proc. of ACL-25.*

König, E. 1990, 'The complexity of parsing with extended categorial grammars', *Proc. of COLING-90.*

König, E. 1991, 'Parsing categorial grammar.' DYANA, deliverable R1.2.C.

Lambek, J. 1958. 'The mathematics of sentence structure.' *American Mathematical Monthly* **65**. 154–170.

Morrill, G. 1989. 'Intensionality, boundedness, and modal logic.' Research Paper EUCCS/RP–32, Centre for Cognitive Science, University of Edinburgh.

Morrill, G. 1990. 'Intensionality and Boundedness', *Linguistics and Philosophy*, **13**.

Morrill, G., Leslie, N., Hepple, M. and Barry, G. 1990. 'Categorial Deductions and Structural Operations', in Barry and Morrill, 1990.

Prawitz, D. 1965. *Natural Deduction: a Proof Theoretical Study*, Almqvist and Wiksell, Uppsala.

Steedman, Mark. 1987. 'Combinatory Grammars and Parasitic Gaps', *NLLT*, **5:3**.

Szabolcsi, A. 1987 'On Combinatory Categorial grammar', *Proc. of the Symposium on Logic and Language, Debrecen,* Akadémiai Kiadó, Budapest.