HAL open science

Graph Algorithms for Improving Type-Logical Proof Search

Richard Moot

To cite this version:

Richard Moot. Graph Algorithms for Improving Type-Logical Proof Search. Categorial grammars - an efficient tool for natural language processing, Jun 2004, Montpellier, France. pp.13-28. hal-00012408

HAL Id: hal-00012408

https://hal.science/hal-00012408v1

Submitted on 15 May 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

hal-00012408, version 1 - 15 May 2008

Graph Algorithms for Improving Type-Logical Proof Search

Richard Moot *

SIGNES, LaBRI
351, cours de la Libération
33405 Talence, France

Abstract

Proof nets are a graph theoretical representation of proofs in various fragments of type-logical grammar. In spite of this basis in graph theory, there has been relatively little attention to the use of graph theoretic algorithms for type-logical proof search.

In this paper we will look at several ways in which standard graph theoretic algorithms can be used to restrict the search space. In particular, we will provide an $O(n^4)$ algorithm for selecting an optimal axiom link at any stage in the proof search as well as an $O(kn^3)$ algorithm for selecting the k best proof candidates.

Key words: Automated Deduction, Floyd-Warshall Algorithm, Lambek Calculus, Proof Net, Ranked Assignments

1 Introduction

Type-logical grammar (Morrill, 1994; Moortgat, 1997) is an attractive logical view of grammatical theory. Advantages of this theory over other formalisms include a simple, transparent theory of (λ term) semantics thanks to the Curry-Howard isomorphism and effective learning algorithms for inducing grammars from linguistic data (Buszkowski and Penn, 1990).

Proof nets, first introduced for linear logic by Girard (1987), are a way of presenting type-logical proofs which circumvents the redundancies present in, for example, the sequent calculus by performing all logical rules 'in parallel'.

* CNRS and INRIA Futurs
Email address: `Richard.Moot@labri.fr` (Richard Moot).
URL: `http://www.labri.fr/Perso/ moot` (Richard Moot).

The only non-determinism in trying to prove a theorem consists of selecting pairs of axiom links. Each possible selection — if correct — will result in a different proof.

However, many of these possible selections can never contribute to a proof net, while a naive algorithm might try these selections many times. It is the goal of this paper to provide algorithms for filtering out these possibilities at an early stage and selecting the axiom link which is most restricted, thereby improving the performance of proof search.

Given that the problem we are trying to solve is known to be NP complete, even in the non-commutative case, it would be too much to hope for a polynomial algorithm (Kanovich, 1991; Pentus, 2003). However, we will see an algorithm for computing the best possible continuation of a partial proof net in $O(n^4)$.

A second aim is to develop a polynomial algorithm by means of approximation. If we consider only the best k axiom links, then we can find these in $O(kn^3)$. When best is defined as 'having axiom links with the shortest total distance' this algorithm converges with results on proof nets and processing (Johnson, 1998; Morrill, 1998).

2 Proof Nets and Essential Nets

In this section we will look at two ways of presenting proof nets for multiplicative intuitionistic linear logic (**MILL**) together with their correctness criteria and some basic properties.

Though the results will be focused on an associative, commutative system, it is simple to enforce non-commutativity by demanding the axiom links to be planar (Roorda, 1991) or by labeling, either with pairs of string positions (Morrill, 1995) or by algebraic terms (de Groote, 1999). In order to have more flexibility in dealing with linguistic phenomena, other constraints on the correctness of proof nets have been proposed (Moot and Puite, 2002), but given that the associative, commutative logic is the worst case (in the sense that it allows the most axiom links) with respect to other fragments of categorial grammars there are no problems adapting the results of this paper to other systems. However, we leave the question of whether it is possible to perform better for more restricted type-logical grammars open.

The choice of presenting the logic with two implications which differ only in the order of the premisses of the links is intended to make the extensions to the non-commutative case more clearly visible.

$$\frac{}{A \vdash A}\,[Ax] \qquad \frac{\Delta \vdash A \quad \Gamma, A, \Gamma' \vdash C}{\Gamma, \Delta, \Gamma' \vdash C}\,[Cut]$$

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \bullet B, \Delta \vdash C}\,[L\bullet] \qquad \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \bullet B}\,[R\bullet]$$

$$\frac{\Delta \vdash B \quad \Gamma, A, \Gamma' \vdash C}{\Gamma, A \,/\, B, \Delta, \Gamma' \vdash C}\,[L/] \qquad \frac{\Gamma, B \vdash A}{\Gamma \vdash A \,/\, B}\,[R/]$$

$$\frac{\Delta \vdash B \quad \Gamma, A, \Gamma' \vdash C}{\Gamma, \Delta, B \setminus A, \Gamma' \vdash C}\,[L\backslash] \qquad \frac{B, \Gamma \vdash A}{\Gamma \vdash B \setminus A}\,[R\backslash]$$

Table 1
The sequent calculus for **L/MILL** with commutativity implicit

$$\frac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{np \vdash np}\,[Ax] \quad \dfrac{}{s \vdash s}\,[Ax]}{np, np \setminus s \vdash s}\,[L\backslash]}{np \vdash s \,/\, (np \setminus s)}\,[R/] \quad \dfrac{}{s \vdash s}\,[Ax]}{np, (s \,/\, (np \setminus s)) \setminus s \vdash s}\,[L\backslash]}{(s \,/\, (np \setminus s)) \setminus s \vdash np \setminus s}\,[R\backslash] \quad \dfrac{}{s \vdash s}\,[Ax]}{s \,/\, (np \setminus s), (s \,/\, (np \setminus s)) \setminus s \vdash s}\,[L/]$$

Fig. 1. Example sequent derivation

2.1 Sequent Calculus

Table 1 shows the sequent calculus for the Lambek calculus **L**, first proposed by Lambek (1958). The commutative version, the Lambek-van Benthem calculus **LP**, is also known as the multiplicative fragment of intuitionistic linear logic **MILL**. An example sequent derivation is shown in Figure 1.

2.2 Proof Nets

Proof nets are an economic way of presenting proofs for linear logic, which is particularly elegant for the multiplicative fragment. When looking at sequent proofs, there are often many ways of deriving essentially 'the same' proof. Proof nets, on the other hand, are inherently redundancy-free.

We define proof nets as a subset of *proof structures*. A proof structure is a collection of the links shown in Table 2 which satisfies the conditions of Definition 1. A link has its conclusions drawn at the bottom and its premisses at the top. The axiom link, top left of the table, has no premisses and two conclusions which can appear in any order. The cut link, top right of the table, is mentioned only for completeness; we will not consider cut links in

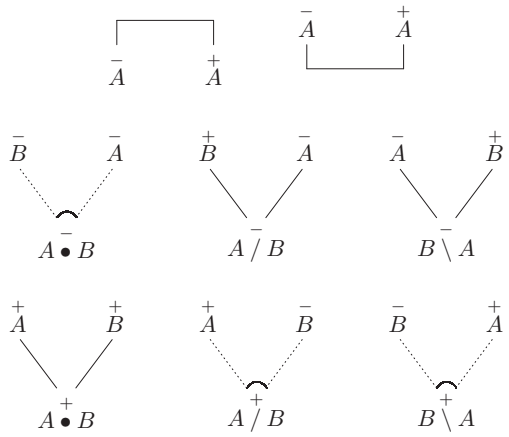


Table 2
Links for proof structures

this paper. A cut link has two premisses, which can appear in any order, and no conclusions. All other links have an explicit left premiss and right premiss. We also distinguish between negative (antecedent) and positive (succedent) formulas and between tensor (solid) and par (dotted) links.

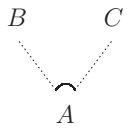
Definition 1 *A proof structure $\mathcal{S}$ is a collection of links such that:*

(1) every formula is the conclusion of exactly one link,
(2) every formula is the premiss of at most one link, formulas which are not the premiss of a link are called the conclusions *of the proof structure,*
(3) a proof structure has exactly one positive conclusion.

Given a proof structure, we want to decide if it is a proof net, that is, if it corresponds to a sequent proof. A correctness criterion allows us to accept the proof structures which are correct and reject those which are not. In this section, we will look at the acyclicity and connectedness condition from Danos and Regnier (1989), which is perhaps the most well-known correctness condition for proof nets in multiplicative linear logic. We will look at another condition in the next section.

Definition 2 *For a proof structure $\mathcal{S}$, a* switching *ω for $\mathcal{S}$ is a choice for every par link of one of its premisses.*

Definition 3 *From a proof structure $\mathcal{S}$ and a switching ω we obtain a* correction graph *$\omega\mathcal{S}$ by replacing all par links*



depending on whether ω selects the left or the right premiss of the link, by one

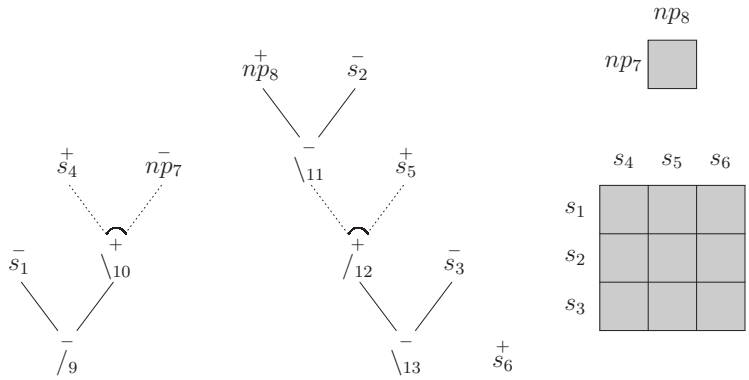
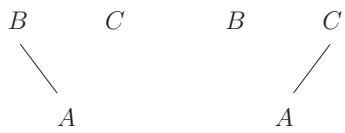


Fig. 2. Example proof frame

of the following links.



Theorem 4 (Danos and Regnier (1989)) *A sequent $\Gamma \vdash C$ is provable in* ***MILL*** *iff all correction graphs of the corresponding proof structure are acyclic and connected, ie. it is a* proof net.

Proof search in a proof net system is a rather direct reflection of the definitions. Given a sequent $\Gamma \vdash C$ we unfold the negative formulas in Γ and the positive formula C, giving us a proof frame. Note that given a polarized formula, exactly one link will apply, making this stage trivial. An example proof frame for the sequent

$$(np \setminus s), (s \,/\, (np \setminus s)) \setminus s \vdash s$$

of Figure 1 is given in Figure 2. We have given the atomic formulas an index as subscript only to make it easier to refer to them; the numbers are not formally part of the logic. The matrix next to the proof frame in the figure represents the possible linkings: the rows are the negative formula occurrences, whereas the columns are the positive formula occurrences. White entries represent currently impossible connections whereas colored entries represent the current possibilities.

The next stage consists of transforming the proof frame into a proof structure by linking atomic formulas of opposite polarity. It is this stage which will concern us in this paper. This is a matter of putting exactly one mark in every row and every column of the table. Figure 3 shows one of the 6 possible

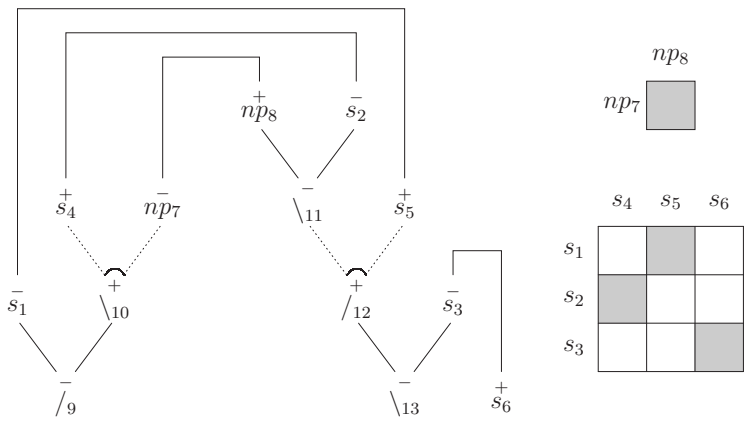


Fig. 3. Example proof net

linkings next to the matrix which corresponds to it.

Finally, we need to check the correctness condition. Though there are many correction graphs for a proof structure, Guerrini (1999) shows we can verify the correctness of a proof structure in linear time. The proof structure in Figure 3 is indeed a proof net, which we can verify by testing all correction graphs for acyclicity and connectedness. Of the 5 alternative linkings, only one other produces a proof net.

2.3 Essential Nets

For out current purposes, we are interested in an alternative correctness criterion proposed by Lamarche (1994). This criterion is based on a different way of decomposing a sequent, this time into a directed graph, with conditions on the paths performing the role of a correctness criterion. A net like this is called an *essential net*. The links for essential nets are shown in Table 3 on the next page, though we follow de Groote (1999) in reversing the arrows of Lamarche (1994).

Definition 5 *Given an essential net $\mathcal{E}$ its positive conclusion is called the* output *of the essential net and the negative conclusions, as well as the negatives premisses of any positive / or \ link, are called its* inputs.

Definition 6 *An essential net is* correct *iff the following properties hold.*

(1) it is acyclic,
(2) every path from the negative premiss of a positive / or \ link passes through the conclusion of this link,
(3) every path from the inputs of the graph passes reaches the output of the graph.

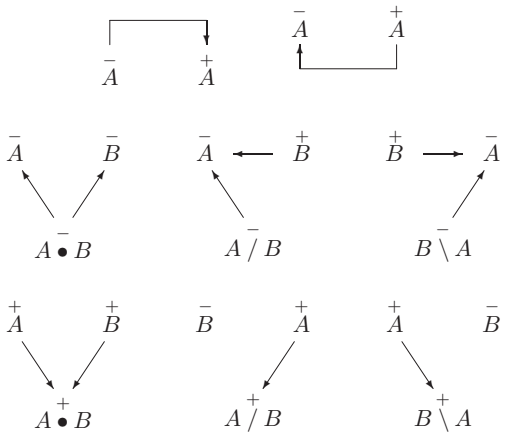


Table 3
Links for essential nets

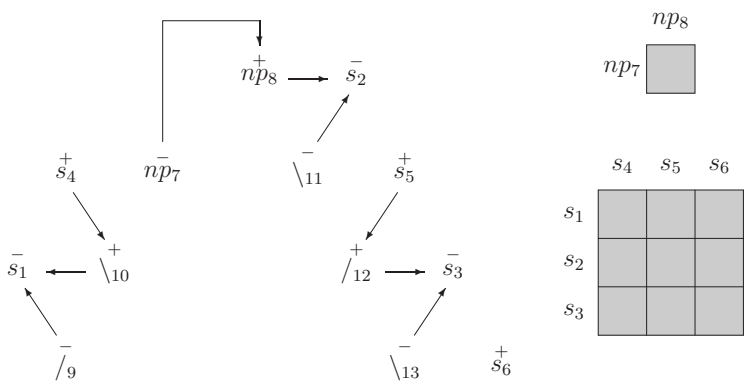


Fig. 4. Example essential net

Theorem 7 (Lamarche (1994)) *A sequent $\Gamma \vdash C$ is provable in **MILL** iff its essential net is correct.*

Condition (1) reflects the acyclicity condition on proof nets, whereas conditions (2) and (3) reflect the connectedness condition. The formulation of 'every path' exists only to ensure correctness of the negative $\bullet$ link; in all other cases there is at most one path between two points in a correct essential net.

Figure 4 gives the essential net corresponding to the proof frame we have seen before, but this time with the *np* axiom link already performed. Remark that we have simply unfolded the formulas as before, just with a different set of links.

It will be our goal to eliminate as many axiom links as possible for this example.

Though the correctness criterion was originally formulated for the multiplicative intuitionistic fragment of linear logic only, Murawski and Ong (2000) show — in addition to giving a linear time algorithm for testing the correctness of

an essential net — that we can transform a classical proof net into an essential net in linear time. So our results in the following sections can be applied to the classical case as well.

2.4 Basic Properties

In order to better analyze the properties of the algorithms we propose, we will first take a look at some basic properties of proof nets.

2.4.1 Axiom Links

Since we will be concerned with finding an axiom linking for a partial proof structure $\mathcal{P}$ which will turn $\mathcal{P}$ into a proof net, we first given some bounds on the number of proof structures we will have to consider. Given that the problem we are trying to solve is NP complete, it is not surprising these bounds are quite high.

Proposition 8 *Let $\mathcal{P}$ be a proof net and f an atomic formula, then the number of positive occurrences of f is equal to the number of negative occurrences of f.*

This proposition follows immediately from the fact that every atomic formula is the conclusion of an axiom link, where each axiom link has one positive and one negative occurrence of a formula f as its conclusion.

Proposition 9 *Every proof frame $\mathcal{F}$ has $O(a!)$ axiom linkings which produce a proof structure, where a is the maximum number of positive and negative occurrences of an atomic formula in $\mathcal{F}$.*

If we have a positive atomic formulas, we have a possibilities for the first one, since all negative formulas may be selected, followed by $a-1$ for the second etc. giving us $a!$ possibilities.

Proposition 10 *Every proof frame $\mathcal{F}$ has $O(4^a)$ planar axiom linkings which produce a proof structure, where a is the maximum number of positive and negative occurrences of an atomic formula in $\mathcal{F}$.*

This follows from the fact that a planar axiom linking is simply a binary bracketing of the atomic formulas and the fact that there are C_{a-1} such bracketings, where C_k, the kth Catalan number, approaches $4^k/\sqrt{\pi}k^{3/2}$.

Proposition 11 *For every partial proof structure with a atomic formulas which are not the conclusion of any axiom link there are $O(a^2)$ possible axiom*

links.

Given that every positive atomic formula can be linked to every negative atomic formula of the same atomic type this gives us a^2 pairs.

2.4.2 Graph Size

Proposition 12 *For every proof structure $\mathcal{S}$ with h negative conclusions, 1 positive conclusion, p par links and t tensor links, the following equation holds.*

$$p + h = t + 1 = a$$

Given Proposition 8, the number of positive and negative atomic formulas is both a. Suppose we want to construct a proof structure $\mathcal{S}$ with h negative conclusions and 1 positive conclusion from these atomic formulas. When we look at the links in Table 2 we see that all par links reduce the number of negative conclusions by 1 and all tensor links reduce the number of positive conclusions by 1.

Proposition 13 *Every essential net $\mathcal{E}$ has $v = h+1+2(t+p) = O(a)$ vertices and $2t + p \leq e \leq 2(t + p) + a = O(a)$ edges.*

This follows immediately from inspection of the links: all conclusions of the essential net (h negative and 1 positive) start out as a single vertex and every link adds two new vertices. For the edges: the minimum number is obtained when we have no axiom links and all par links are positive links for $\backslash$ or $/$ which introduce one edge, the maximum number includes a axiom links and par links which are all negative links for $\bullet$.

Corollary 14 *An essential net is* sparse*, ie. the number of edges is proportional to the number of vertices, but if we add edges for all possible axiom links it will be* dense*, ie. e is proportional to v^2,*

Immediate from Proposition 11 and Proposition 13.

3 Acyclicity

We begin by investigating the acyclicity condition, condition (1) from Definition 6, which is the easiest to verify.

In order to select the axiomatic formula which is most constrained with respect to the acyclicity condition we can simply enumerate all a^2 possible axiom links

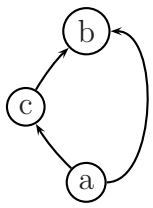


Fig. 5. Eliminating node c from the path from a to b

and reject those which produce a cycle.

We can easily verify whether a graph contains a cycle in time proportional to the representation of the graph, $v + e$, using either breadth-first search or depth-first search (e.g. Cormen et al., 1990, Section 23.2 and 23.3 respectively), giving us an $O(a^2(v + e)) = O(a^3)$ algorithm for verifying all pairs.

However, this means we will visit the vertices and edges of the graph many times. It is therefore a practical improvement to compute the transitive closure of the graph in advance, after which we can perform the acyclicity queries in constant time.

In this paper we will use the Floyd-Warshall algorithm (Cormen et al., 1990) for computing the transitive closure, which computes the transitive closure of a directed graph in $O(v^3)$ time. Though there are algorithms which perform asymptotically better for sparse graphs, it is hard to beat this algorithm in practice even for sparse graphs because of the small constants involved, while for dense graphs, which we will consider in the next section when we take all possible axiom links into account, it is the algorithm of choice (Sedgewick, 2001).

The Floyd-Warshall algorithm is based on successively eliminating the intermediate vertices c from every path from a to b. Given a vertex c and the paths $a \to c \to b$ for all a and b we create a direct path $a \to b$ if it didn't exist before. That is to say, there is a path from a to b if either there is a path from a to c and from c to b or if there is a path from a to b which we already knew about (Figure 5).

$$path(a, b) := path(a, b) \vee (path(a, c) \wedge path(c, b)) \tag{1}$$

After eliminating c, for every path in the original graph which passed through c there is now a shortcut which bypasses c. After we have created such shortcuts for all vertices in the graph it is clear that the resulting graph has an edge $a \to b$ iff there is a path from a to b in the original graph.

Figure 6 on the next page shows the essential net of Figure 4 on page 7 in adjacency matrix representation (left of the figure) and its transitive closure

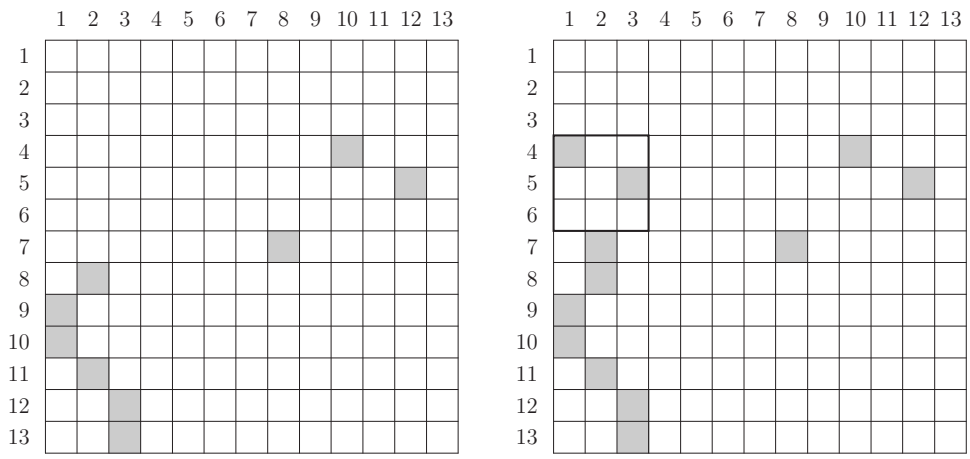


Fig. 6. Initial graph (left) and its transitive closure (right)

(right of the figure). A square in the matrix is colored in iff there is a link from the row to the column in the graph.

The relevant part of the graph for the acyclicity test is marked by a square around columns $1-3$ of row $4-6$. We see here, for example, that given the existence of a path $4 \to 1$ an axiom link between s_1 and s_4 would produce a cycle (via node 10 in the original graph) and is therefore to be excluded. A similar observation can be made for s_5 and s_3.

4 Connectedness

Verifying conditions (2) and (3) from Definition 6 is a bit harder. The question we want to ask about each link is: does this link contribute to a connected proof structure? Or, inversely, does excluding the other possibilities for the two atomic formulas we connect mean a connected proof structure is still possible.

To check the conditions we need to verify the following:

(1) for every negative input of the net we verify there exists a path to the positive conclusion,
(2) for every negative $\bullet$ link we verify that *both* paths leaving from it reach their destination,
(3) for every positive / or \ link we check the existence of a path from its negative premiss to its positive conclusion continuing to the positive conclusion of the essential net.

Given that we are already computing the transitive closure of the graph for verifying acyclicity, we can exploit this by adding additional information to the

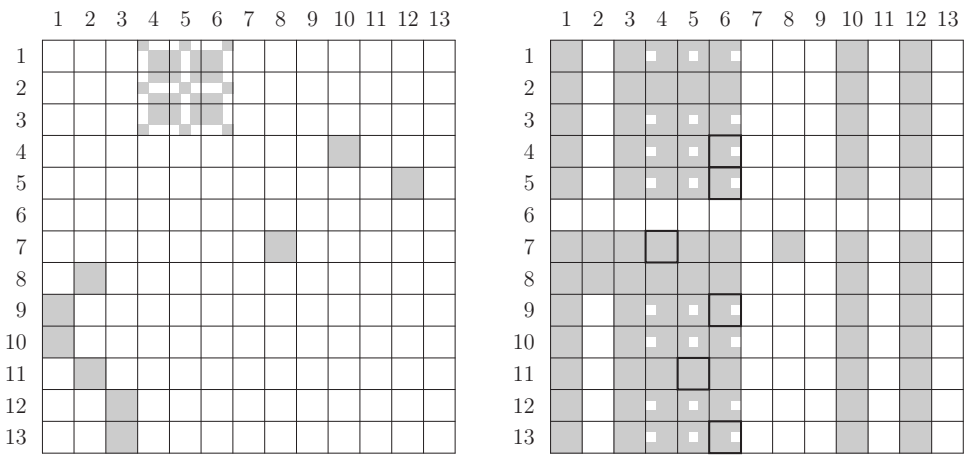


Fig. 7. Initial graph (left) and its transitive closure (right)

matrix we use for the transitive closure. There are many ways of storing this extra information, the simplest being in the form of an ordered list of pairs. Given that, for a atomic formulas, each possible connection allows $(a-1)^2$ other connections (ie. it is agnostic about all possibilities not contradicting this one) but excludes $2(a-1)$ possibilities, it is more economic to store the connections which are excluded. For example, the ordered set associated to the edge from 1 to 4 will be $\{1-5, 1-6, 2-4, 3-4\}$, meaning "there is an edge from 1 to 4 but not to anywhere else and the only edge arriving at 4 comes from 1".

Note that in the description of the Floyd-Warshall algorithm, we made use only of the logical 'and' and 'or' operators. For ordered sets, the corresponding operations are set union and set intersection. For eliminating vertex c from a path from a to b, we first take the union of the ordered set representing the links which are not in a path from a to c with that representing the links not in a path from c to b (any vertex in either set couldn't be in a path from a via c to b). Then, we take the intersection of this set with the old set associated to the path from a to b.

$$\overline{path}(a,b) := \overline{path}(a,b) \cap (\overline{path}(a,c) \cup \overline{path}(c,b)) \tag{2}$$

Note that Equation 2 is simply Equation 1 with both sides negated, the negations moved inward and set union and intersection in the place of the logical 'or' and 'and' operators.

Given that we can implement the union and intersection operations in linear time with respect to the size of the input sets, the total complexity of our algorithm becomes $O(v^3 2(a-1)) = O(a^4)$.

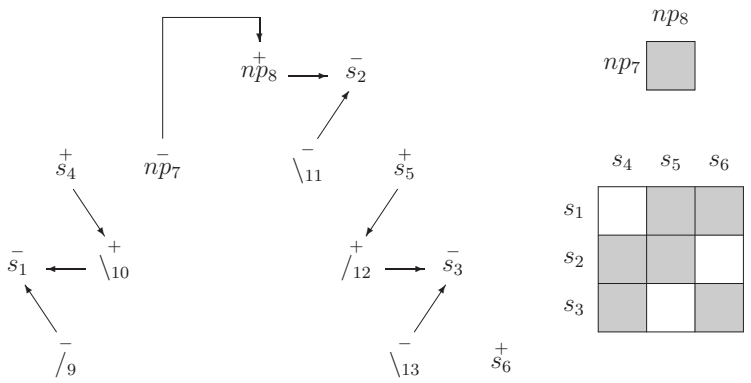


Fig. 8. Essential net with acyclicity and connectedness taken into account

Figure 7 on the page before shows the initial graph and its transitive closure. Every entry from the previous graph is now divided into 9 subentries — one for each possible axiom link. We read the entry for $1-4$ as follows: the first row indicate the link between 1 and 4, and thereby the absence of a link $1-5$ and $1-6$, the second row indicates the possibilities for linking 2, which just excludes $2-4$ and the third row indicates that for 3 just the $3-4$ connection is impossible. Again, we have marked the table entries relevant for our correctness condition by drawing a black border around them.

When we look at the transitive closure, we see that, should we choose to link s_2 to s_6, this would make it impossible to reach vertex 6 from vertices 4, 5, 9 or 13. Remark also that cycles need to be excluded separately. For example, the path from 5 to 1 in Figure 7 does not mean we need to exclude the $s_1 - s_5$ axiom link.

Figure 8 shows the proof frame of Figure 4 with all constraints taken into account. We see that, whatever choice we make for the first axiom link, all other axiom links will be fixed immediately, giving us the two proofs $s_1 - s_5$, $s_2 - s_4$, $s_3 - s_6$ (ie. the proof shown in Figure 3) and $s_1 - s_6$, $s_2 - s_5$, $s_3 - s_4$.

5 Extensions and Improvements

An interesting continuation of the themes explored in this paper would be to look at *dynamic* graph algorithms, where we maintain the transitive closure under additions and deletions of edges. This would avoid recomputing the transitive closure from scratch after every axiom link and would allow us to take advantage of the information we have already computed. King and Saggert (1999) propose an algorithm with $O(n^2)$ update time based on keeping track of the number of paths between two vertices, which is easy enough to adapt to our current scenario in the case of acyclicity tests, though it remains unclear if it can be adopted to check for connectedness.

Another improvement would be to represent the ordered sets differently. Given that their structure is quite regular, it may be possible to improve upon linear time union and intersection. However, given that for each iteration the size of the sets either remains the same or decreases (the principal operation in Equation 2 being intersection), it remains to be seen if this will result in a practical improvement.

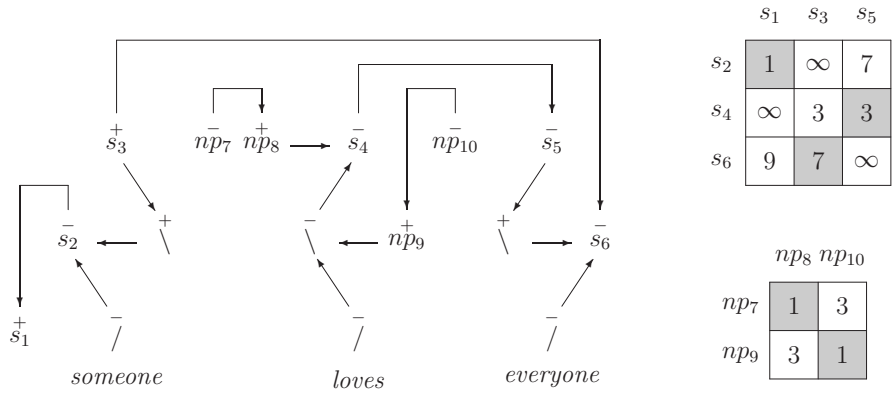
Finally, we can consider the work in the two previous sections as using a sort of 'lookahead' of one axiom link, which is to say we exclude all axioms links which, by themselves, would produce a cyclic or disconnected proof structure. This can be extended quite naturally to doing k axiom links at the same time, though each extra axiom link will multiply the required space by $O(n^2)$, the required time for acyclicity by $O(n^3)$ and the required time for connectedness by $O(n^4)$ (this is relatively easy to see because we are in effect substituting $n(n-1)$ for the old value of n).

6 Polynomial Time

If we add weights to the different connections, the situation changes. The simplest way to add weights to our graph is to use the *distance* between two atomic formulas as their weight and prefer the total axiom linking with the least total weight. This choice of assigning weights is closely related to work on left-to-right processing of sentences, proposed independently by Johnson (1998) and Morrill (1998). The claim they make is that the complexity of a phrase depends on the number of 'open' or unlinked axiom formulas a reader/listener will have to maintain in memory to produce a parse for this sentence.

Finding a minimum-weight solution to this problem is known as the *assignment problem*. Murty (1968) was the first to give an algorithm for generating the assignments in order of increasing cost. His $O(kn^4)$ algorithm for finding the k best assignments can be improved to $O(kn^3)$, even though tests on randomly generated graphs have shown the observed complexity to be $O(kn^2)$ (Miller et al., 1997).

Because using the distance as weight tends to favor cyclic connections, it is preferable to make one pass of the algorithm described in the previous section and assign a weight of infinity to all edges which are either cyclic or disconnected. Figure 9 shows a weighted graph corresponding to an example from Morrill (1998), the sentence 'someone loves everyone', which has a preferred reading where the subject has wide scope. Of the four readings of this sentence (two if we enforce planarity) there is a preference for connecting $s_1 - s_2$, $s_3 - s_6$, $s_5 - s_4$, with a total weight of 11, as compared to connecting $s_1 - s_6$, $s_3 - s_4$, $s_5 - s_2$, with a total weight of 19.



	s_1	s_3	s_5
s_2	1	∞	7
s_4	∞	3	3
s_6	9	7	∞

	np_8	np_{10}
np_7	1	3
np_9	3	1

Fig. 9. Minimum Weight Linking for 'someone loves everyone'

There is an important difference between using distance weights like we do here and keeping track of the open axiom links like Johnson and Morrill, which is that we select a linking which is best *globally*. It is therefore to be expected that we will make different predictions for some 'garden path' sentences (ie. sentences where a suboptimal local choice for the axiom links will be made).

Finding an appropriate value of k and fragments of type-logical grammar for which this k is guaranteed to find all readings remains an interesting open question. Cautious people might select $k = n!$, given that in any type-logical grammar there are at most $n!$ links possible, to generate all readings in increasing order of complexity. It seems tempting to set k to n^3, because many interesting grammar formalisms have $O(n^6)$ complexity (Joshi et al., 1991), and find type-logical fragments for which we can show proof search using this strategy is complete.

7 Conclusion

We have seen how standard graph algorithms can be modified to aid in proof search for type-logical grammars by rejecting connections which can never contribute to a successful proof.

We have also seen how weighing the connections allows us to enumerate the links in increasing order and linked this with processing claims.

References

Buszkowski, W., Penn, G., 1990. Categorial grammars determined from linguistic data by unification. Studia Logica 49, 431–454.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., 1990. Introduction to Algorithms. MIT Press, Cambridge, Massachusetts.

Danos, V., Regnier, L., 1989. The structure of multiplicatives. Archive for Mathematical Logic 28, 181–203.

de Groote, P., 1999. An algebraic correctness criterion for intuitionistic proof-nets. Theoretical Computer Science 224 (1–2), 115–134.

Girard, J.-Y., 1987. Linear logic. Theoretical Computer Science 50, 1–102.

Guerrini, S., 1999. Correctness of multiplicative proof nets is linear. In: Fourteenth Annual IEEE Symposium on Logic in Computer Science. IEEE Computer Science Society, pp. 454–263.

Johnson, M., 1998. Proof nets and the complexity of processing center-embedded constructions. Journal of Logic, Language and Information 7 (4), 443–447.

Joshi, A., Shanker, V., Weir, D., 1991. The convergence of mildly context-sensitive grammar formalisms. In: Sells, P., Shieber, S., Wasow, T. (Eds.), Foundational Issues in Natural Language Processing. MIT Press, Cambridge, Massachusetts, pp. 31–82.

Kanovich, M., 1991. The multiplicative fragment of linear logic is NP-complete. Tech. rep., University of Amsterdam, ITLI Prepublication Series X-91-13.

King, V., Saggert, G., 1999. A fully dynamic algorithm for maintaining the transitive closure. In: Proceedings of the 31st Annual ACM Symposium on the Theory of Computing. pp. 492–498.

Lamarche, F., 1994. Proof nets for intuitionistic linear logic I: Essential nets. Tech. rep., Imperial College.

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

Miller, M. L., Stone, H. S., Cox, I. J., 1997. Optimizing Murty's ranked assignment method. IEEE Transactions on Aerospace and Electric Systems 3 (33), 851–865.

Moortgat, M., 1997. Categorial type logics. In: van Benthem, J., ter Meulen, A. (Eds.), Handbook of Logic and Language. Elsevier/MIT Press, Ch. 2, pp. 93–177.

Moot, R., Puite, Q., 2002. Proof nets for the multimodal Lambek calculus. Studia Logica 71 (3), 415–442.

Morrill, G., 1994. Type Logical Grammar: Categorial Logic of Signs. Kluwer Academic Publishers, Dordrecht.

Morrill, G., 1995. Higher order linear logic programming of categorial deduction. In: Proceedings of the European Association for Computational Linguistics. Dublin, pp. 133–140.

Morrill, G., 1998. Incremental processing and acceptability. Tech. Rep. LSI–98–46–R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya.

Murawski, A. S., Ong, C.-H. L., 2000. Dominator trees and fast verification of proof nets. In: Logic in Computer Science. pp. 181–191.

Murty, K. G., 1968. An algorithm for ranking all the assignments in order of increasing cost. Operations Research 16, 682–687.

Pentus, M., 2003. Lambek calculus is NP-complete. Tech. rep., CUNY.

Roorda, D., 1991. Resource logics: A proof-theoretical study. Ph.D. thesis, University of Amsterdam.

Sedgewick, R., 2001. Algorithms in C: Graph Algorithms. Addison Wesley.