Proof structures of a fragment of $NL\Diamond_{\mathscr{R}}$ as HRG

René van Gasteren

July 15, 2012

Abstract

In this paper I show that the proof of Moot(2008) showing that $NL\Diamond_{\mathscr{R}}$ and TAGs are equivalent, is only correct for a fragment of $NL\Diamond_{\mathscr{R}}$. Also we will see how a proof net and proofstructure of such a fragment can be made using Hyperedge Replacement Grammars (HRGs). Since HRGs are context free graph grammars, I will also show a way to compile away the structural rules.

1 Introduction

Moot(2008) shows that both $NL\Diamond_{\mathscr{R}}$ and LG generate the same class of languages as TAGs, using hyperedge replacement grammars as an intermediate step. Thereby showing that $NL\Diamond_{\mathscr{R}}$ and LG are mildly context-sensitive formalisms and therefor benefit from the property of polynomial parsability.
We're going to see in this paper that this is only correct for a fragment of $NL\Diamond_{\mathscr{R}}$ and LG Grammar. The former needs a restriction on the use of the structural rules in $\mathscr{R}$, the latter is proven to be more expressive than TAG. Moot himself gives a solution for this problem, although not explicitly in his paper. I will use a 2008 presentation and a draft called 'Type-logical and Hyperedge Replacement grammars', both work by Moot, to give a solution for $NL\Diamond_{\mathscr{R}}$
I will first, in section 2, introduce Hyperedge Replacement Grammar. This grammar plays a central role, because it is used to represent proof nets. In section 3, I will show the relations between different grammars like Moot did in his paper, and show that there is a problem with it. In section 4, I will try to show how to deal with this problem and give a good example of proof nets as HRG.

2 Hyperedge Replacement Grammars

A Hyperedge Replacement Grammar (HRG) is a graph grammar where a language is built by replacing hyperedges for hypergraphs. HRG's were introduced by Bauderon & Courcelle(1987) and Habel & Kreowski(1987) as a type of context free graph grammar. I'll give a definition of hypergraphs, hyperedge replacement and hyperedge replacement grammars below.

Definition 1.1 A hypergraph generalizes the notion of graph by allowing the edges, called hyperedges, to connect not just two but any number of nodes. Let Γ be an alphabet edge labels and let σ be an alphabet of selectors. A hypergraph over Γ and σ is a tuple $\langle V, E, lab, nod, ext\rangle$, where

V is the finite set of vertices,

E is the finite set of hyperedges disjoint with V,

lab is a labeling function, from E to Γ, assigning an edge label to each hyperedge,

nod is the incidence function that associates with each edge $e \in E$ a partial function $nod(e) : \sigma \to V$, that is, it selects a vertex for every selector σ of the edge,

ext is the external function, a partial function from σ to V, that is for every selector σ of the hypergraph it selects a vertex.

Definition 1.2 The type of a hypergraph H is the domain of the external function, $type(H) = dom(ext)$ The type of an edge e is the domain of the incidence function $type(e) = dom(nod(e))$.

Definition 2 The operation of hyperedge replacement replaces a hyperedge by a hypergraph H of the same type. Let H and K be two disjoint hypergraphs with the same set of edge labels Γ and the same set of selctors σ. Let e be an edge of H such that type(e) = type(K). The hyperedge replacement of e by G, $H[e := G] = \langle V, E, lab, nod, ext\rangle$ is defined as follows.

$V = V_H \cup V_K$,

$E = (E_H - e) \cup E_k$,

$lab = lab_H \cup lab_K$,

$nod = nod_H \cup nod_K$,

$ext = ext_H$,

$Forall s \in type(e), nod_H(e, s) = ext_K(s)$

Definition 3.1 A hyperedge replacement grammar (HRG) is a tuple $G = \langle N, T, \sigma, P, S\rangle$, such that

N is the alphabet of nonterminal edge labels,

T is the disjoint alphabet of terminal edge labels,

σ is the alphabet of selectors,

P is the set of productions,

$S \in N$ is the start nonterminal symbol

Definition 3.2 Let G be a hyperedge replacement grammar. The language generated by G is the set of hyperedges without hyperedges labeled by nonterminal edge labels derivable from S.

Definition 3.3 The rank of a terminal or nonterminal symbol is the number of its tentacles. The rank of a hyperedge replacement grammar is the maximum rank of a nonterminal symbol in the grammar

3 Relations between grammars

Moot(2008) gives 4 lemma's with proof sketches about the relations between $LTAG_{nf}$, $NL\diamondsuit_{\mathscr{R}}, LG, TAG$, and HR_2. Here, I'm just going to mention three of them without proof, because we're assuming these lemma's are correct. Further I will clarify the context and notation of the lemma's. The fourth lemma is the controversial one and is in our focus right now.

Lemma 1: HR_2 grammars generating trees and TAG grammars are strongly equivalent. The lowercase '2' indicates the rank of the hyperedge grammar.

Lemma 2: For every $LTAG$ grammar G there is a weakly equivalent $LTAG_{nf}$ grammar G'. It is also know that every TAG has a strongly equivalent $LTAG$ en vice versa. The criteria of a LTAG in normal form (nf) is not relevant here.

Lemma 3: If G is an $LTAG_{nf}$ grammar, then there is a strongly equivalent $NL\diamondsuit_{\mathscr{R}}$ grammar G' and a strongly equivalent LG grammar G''.

Lemma 4: If G is a Lambek Grammar (specifically $NL\diamondsuit_{\mathscr{R}}$ or LG), then there is a strongly equivalent HR_2 grammar G'.

Figure 1 gives a representation of the relations between the grammars. LG grammar has to be included at the right top.

The fourth lemma is underspecified by Moot. It is not described which structural rules are included in $\mathscr{R}$. In order for this figure and lemma 4 to be correct, the gram-

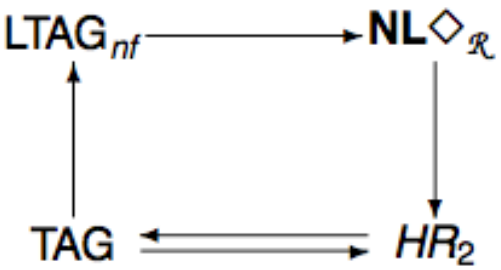


Figure 1: A visual representation of the four lemma's

mars $NL\Diamond_{\mathscr{R}}$ and LG must be referring to a fragment of them. There are proves that $NL\Diamond_{\mathscr{R}}$ and LG grammar are more expressive than TAG. Moot(2002) shows that a large freedom in structural rules in $\mathscr{R}$ gives a PSPACE formalism generating exactly the context-sensitive languages. De Groote(1999) and Capelletti(2007) show that $NL\Diamond_{\mathscr{R}}$ with $\mathscr{R} = \emptyset$ ($NL\Diamond_{\emptyset}$ or just $NL\Diamond$) generates exactly the context-free languages and is polynomial parsable. Hence, the set of structural rules in $\mathscr{R}$ for the grammar the lemma is talking about, is restricted but not empty. Further Mellissen(2009) proves that LG can generate languages, that TAG cannot. In the next section we are going to find out what restrictions are needed in order to make lemma 4 correct for $NL\Diamond_{\mathscr{R}}$.

4 Analysis of structural rules for $NL\Diamond_{\mathscr{R}}$

Unrestricted structural rules give rise to a context-sensitive language and PSPACE decision problem. For a rule the number of links on the left hand side can be more than one, which means context is important. Hyperedge Replacement Grammars are context free graph grammars (Engelfriet 1997). The right hand side of a rule can be any complex graph, the left hand side is always a single nonterminal hyperedge.

To avoid that our combinations of structural rules fall into an associative, commutative system, we will end up with a number of restrictions on the final grammar. This gives us the tools to describe the generated proof nets as a hyperedge replacement grammar, where the structural rules are factored out.

Definition 4.1 Given a structural rule r with connectives $c_1, ..., c_n$ on the left hand side and connectives $d_1, ..., d_n$ on the right hand side a trace is a bijection associating each element c_i with an element of d_j.

Definition 4.2 Let P be a proof net. It is well-bracketed iff there are no two contractions c_1 and c_2 in the conversion sequence of A such that such that on any of the paths between the two traces t_{1a} and t_{1b} of c_1 there is just one of the traces of c_2.

The combination of well-bracketedness and traces guarantee that every path in a proof net can be described by a context free grammar.

Definition 4.3 Given a grammar G with a set of structural rules R, a mode m is inert with respect to this grammar and these structural rules if we can apply all its contraction before applying any structural rules.

Lemma 5 An $NL\diamondsuit_{\mathscr{R}}$ or LG grammar G is well-bracketed if 1) all its external modes are inert, and 2) the structural rules are a subset of one of the following sets $\{P1, P2, P3, P4\}$, $\{P1', P2'\}$, $\{P3', P4'\}$, $\{Gr1, Gr2, Gr3, Gr4\}$, $\{Gr1', Gr2'\}$, $\{Gr3', Gr4'\}$.

In his draft, Moot gives an HR_2 Grammar. This grammar constructs proof structures and proof nets in a way that is very similar to contructing regular proof nets, except for the structural rules, which are compiled away. In the appendix I will show an example of a proof net for 'agent that Trinity escaped from', both in regular style proof nets and in HRG style. The HRG I use is the one in the appendix of Moot 2008 draft.

Let me make some things clear about the idea and representation of proof nets and proof structures as HRG. S is of type $\emptyset$ and is the start, T is a proof net contracting to a tree, and V is a proof net contracting to a vertex. With 'compiling away' in this example we mean that the 'np' in the subformula $\diamondsuit\Box np$ is placed on his position in one step,

$$\frac{\Gamma[\Delta \cdot np] \Rightarrow A}{\Gamma[\Delta] \Rightarrow A/\diamondsuit\Box np}$$

instead of multiple small steps,

$$\cfrac{\cfrac{\cfrac{\vdots}{\Gamma[\Delta \cdot \langle\Box np\rangle] \Rightarrow A}}{\Gamma[\Delta \cdot \diamondsuit\Box np] \Rightarrow A}}{\Gamma[\Delta] \Rightarrow A/\diamondsuit\Box np}$$

In appendix A I show how we derive a proof net from a proofstructure for $n \circ ((n\backslash n)/(s/\diamondsuit\Box np) \circ (np \circ (np\backslash s)/np)) \vdash n$

In appendix B I show how we derive a (abstract) proof structure for the same $NL\diamondsuit_{\mathscr{R}}$ formula, only this time using HR_2 rules. The steps taken are not necessarily the same, and the other way around compared the constructions in A. Also, in this example the proof net is not derived from the proof structure. I have numbered the conversions, such that I can motivate them here. As the reader can see the tensor tree is not complete before we use the 'structural use with unary control' in (4); the last tensor rule is (6). This is why we cannot derive the proof net and the proof structure in the same

derivation. I restructured the structure without using a actual rule in (5). After (6) the proofstructure is complete and I continue with the axiom rules and rewriting T- and V-labeled edges a vertex in (7) (8) (9), ending in the lexical unfolding of the words. As you can see, I do multiple steps at once in (9).

5 Conclusion

We have seen how Moot comes to his conclusion that $NL\Diamond_{\mathscr{R}}$ and LG generate the same class of languages as TAGs. I mentioned several proofs for the claim that this conclusion is only correct for a fragment of both grammars. After I have introduced HRGs as a context free graph grammar, we have seen how to use them as an intermediate step showing the equivalence between TAGs and a fragment of $NL\Diamond_{\mathscr{R}}$, describing proof nets and proof structures as HRG.

6 Bibliography

Melissen, M. The generative capacity of the Lambek-Grishin calculus: A new lower bound. In P. de Groote, M. Egg, and L. Kallmeyer, editors, Proceedings 14th Conference on Formal Grammar, volume 5591 of Lecture Notes in Computer Science, pages 118132. Springer, 2010.

Moortgat, M. & Moot, R. (2012), proof nets and the categorial flow of information.

Moot, R. & Puite, Q. (1999), Proof Nets for Multimodal Categorial Grammars, in Geert-Jan M. Kruijff & Richard T. Oehrle, eds., 'Proceedings of Formal Grammar', Chapter 10.

Moot, Richard and Quintijn Puite. 2002. Proof nets for the multimodal Lambek calculus. Studia Logica, 71(3):415442.

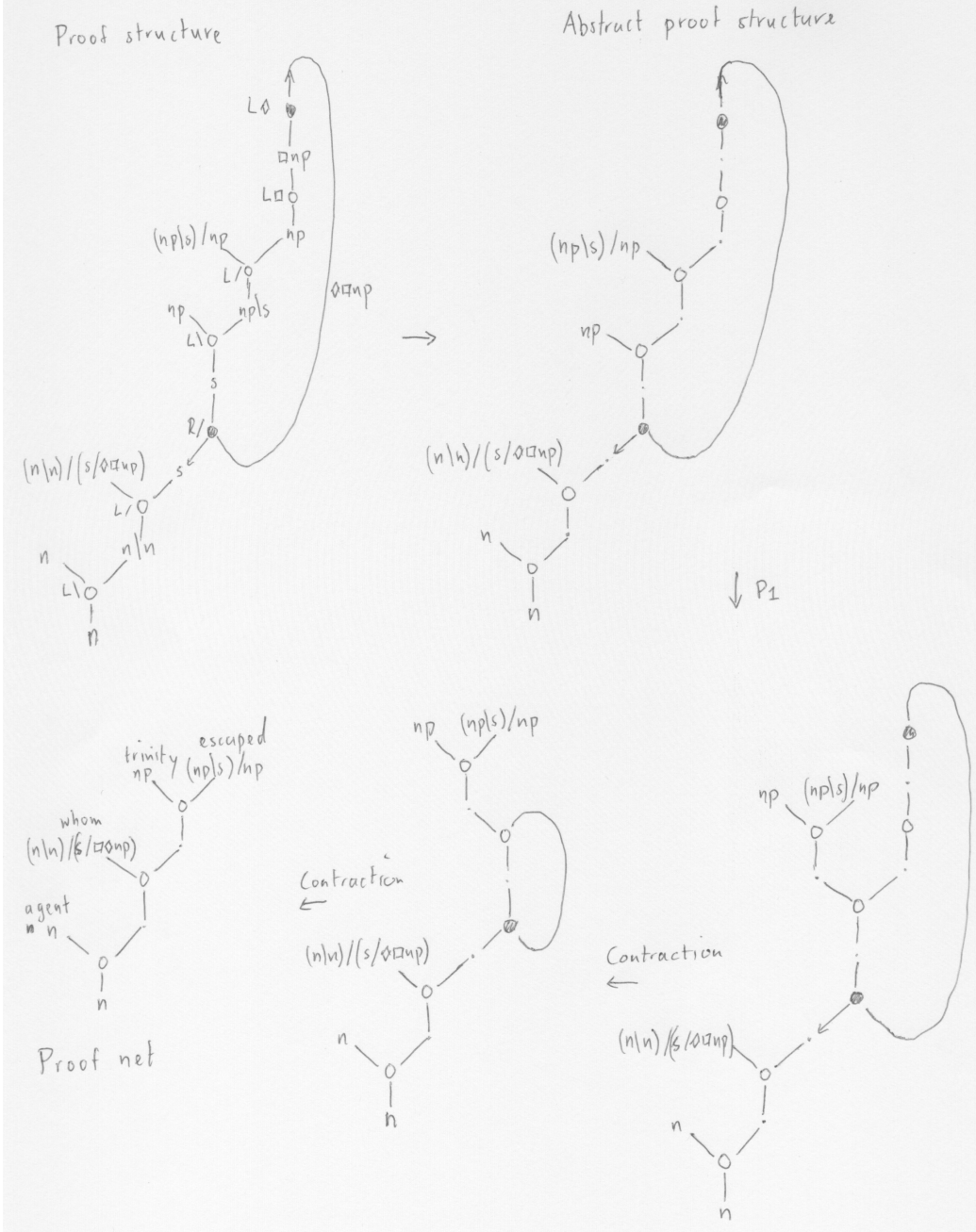
Moot, R. (2008), Type-Logical and Hyperedge Replacement Grammars Draft

7 Appendix

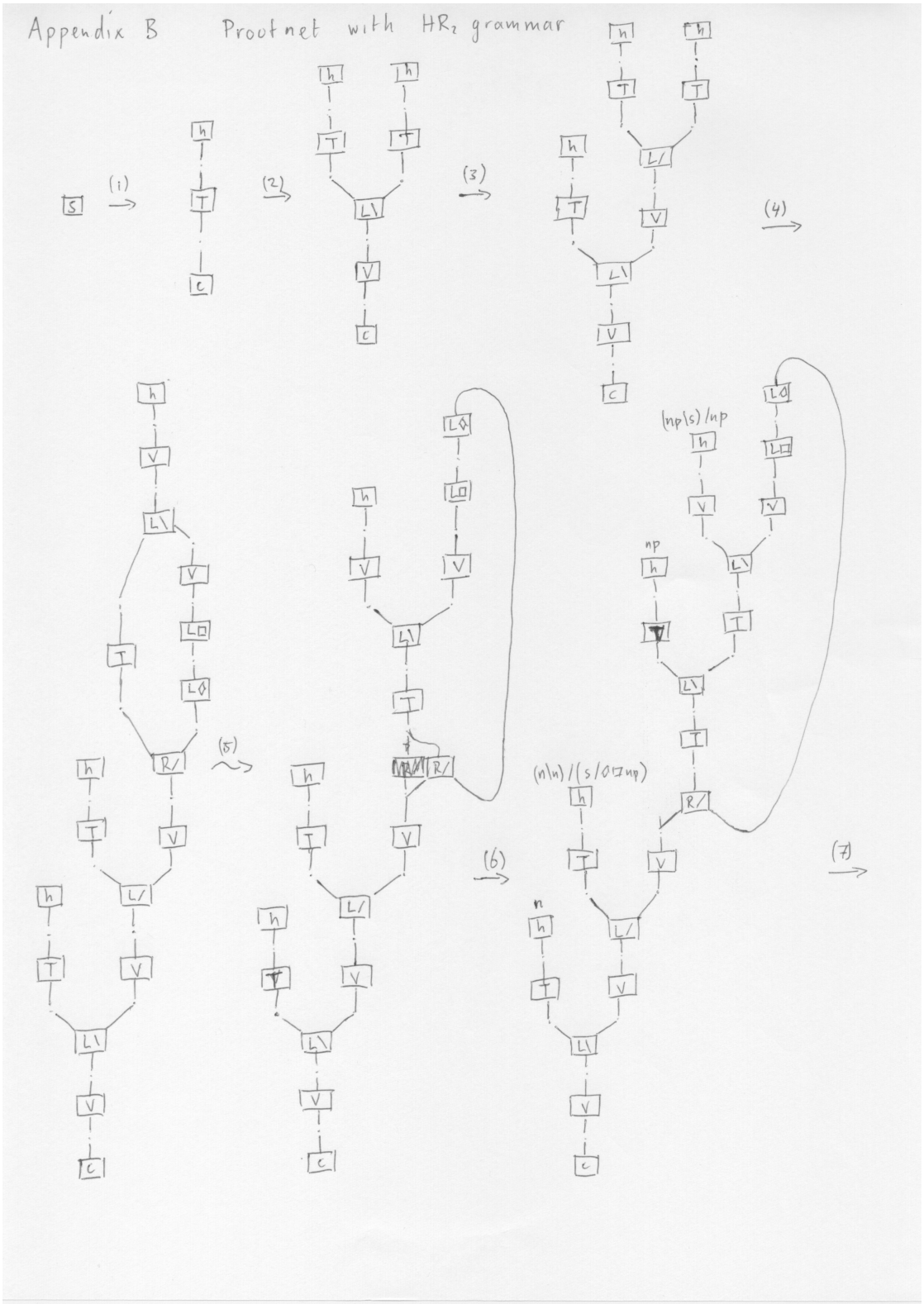
Appendix A Regular proof net

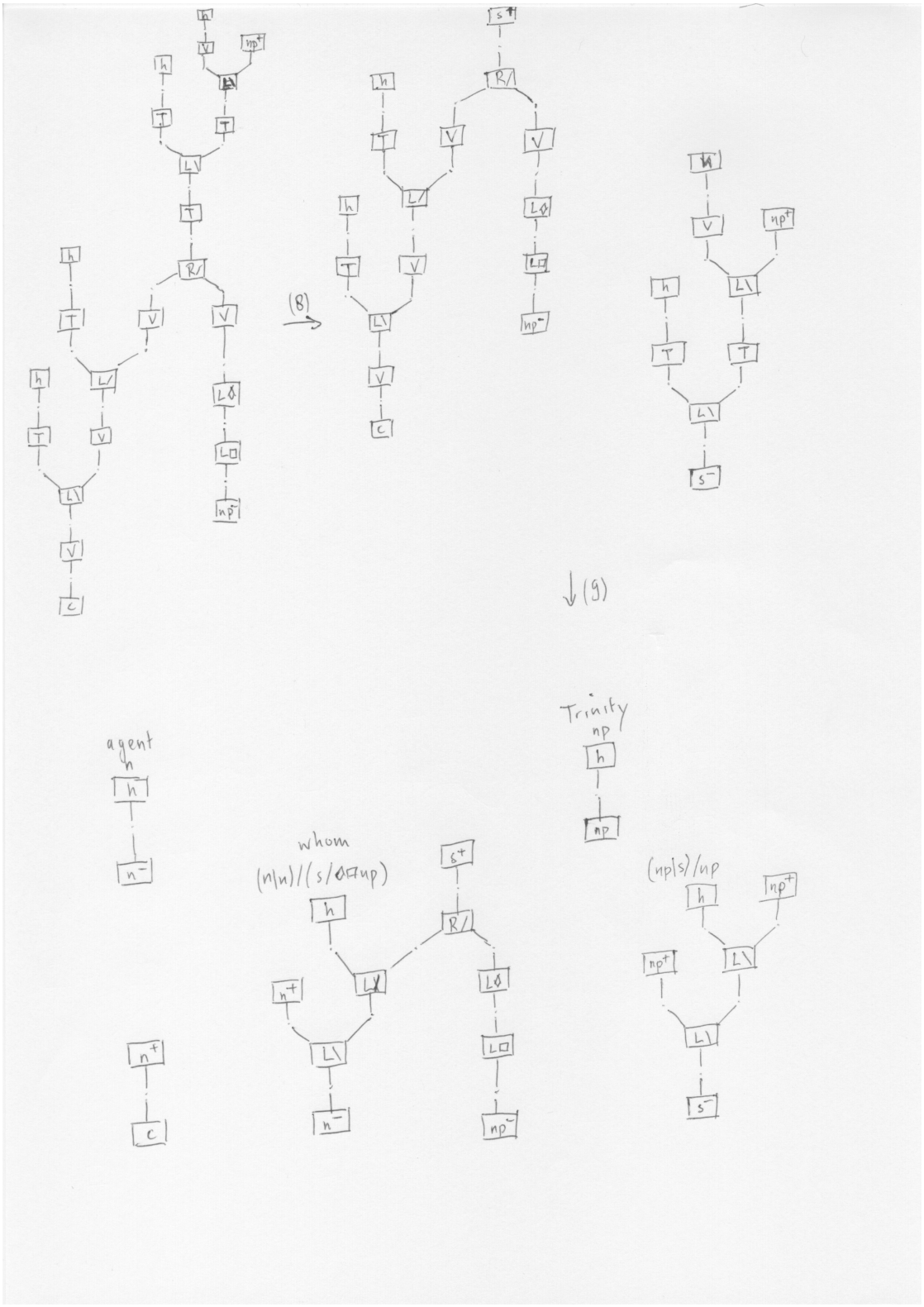
agent whom Trinity escaped ⊢ n

$n \circ ((n\backslash n)/(s/\Diamond\Box np)) \circ (np \circ ((np\backslash s)/np)) \vdash n$



Appendix B Proofnet with HR_2 grammar




(8)
(9)
agent
n
whom
(n|n)/(s/□up)
Trinity
np
(np|s)/np