The CatLog2 Technical Manual

Glyn V. Morrill

Departament de Ciències de la Computació
Universitat Politècnica de Catalunya

25th May 2018

Preface

CatLog is a series of Prolog programs for automatic logical semantic parsing (the computer translation of human language into logic conserving logical form) developed over the last 30 years by Glyn V. Morrill. CatLog2 is a system for such semantic parsing essentially comprising the Prolog program CatLog version f9.1.1 which also outputs the proofs of such prosodic form/semantic form relations: syntactic structures.

The CatLog programs work by reducing grammar to logic: an expression α is of the grammatical type A if and only if an associated logical statement $\Gamma_\alpha \Rightarrow A$ is a theorem of a universal categorial type calculus. This type calculus is an intuitionistic sublinear logic. That it is *sublinear* is because in it the rule of *permutation does not freely preserve theoremhood* (word order matters in grammar). That it is intuitionistic means that its proofs are *constructive*: the meaning of an expression analysed is a function of the meanings of its words and of the constructive content of the proof by which the expression is shown to be grammatical. I.e. the design entails *compositionality*, commonly articulated as the principle that the meaning of a sentence is a function of the meanings of its words and of their mode of composition; a principle generally attributed to the German philosopher G. Frege a century ago (although he never stated it as such). Because grammar is reduced to logic parsing is theorem-proving (Pereira and Warren 1983[7]).

Contents

Chapter 1

Introduction

The overall architectural design is purely lexical: a grammar is just a lexicon which is a particular set of triples of the form:

lex(Prosodic form, Type, Logical form)

Such grammar is called type logical categorial grammar, logical categorial grammar, type logical grammar, or simply logical grammar or categorial grammar; see e.g. the book Morrill (2011[6]). In this architecture the categorial type calculus is universal and a particular language is defined by a lexicon. With respect to the continuous and discontinuous units of Lambek (1988[1]) and Morrill, Valentín, and Fadda (2011[5]), and the bracket modalities of Morrill (1992[3]) and Moortgat (1996[2]), we require that every lexicon be

- **Unit non-negative**: i.e. in lexical types there must be no continuous nor discontinuous unit in antecedent position.
- **Bracket non-negative**: i.e. in lexical types there must be no antecedent position bracket modality nor succedent position antibracket modality (Morrill and Valentín 2015[4]).

If the principles of unit and bracket non-negativity are not respected it is possible for a prosodic form to be associated with an infinite number of logical forms, something which would obviously make semantic parsing listing logical forms non-computable, and something which is not attested empirically. Rather, human language appears to have the *finite reading property* of van Benthem (1991[8]): that in each type an expression is associated with at most a finite number of logical forms.

CatLog2 is essentially the file f9.1.1.pl which contains a logical grammar of English represented by a lexicon, and a set of examples of expressions to be analysed in given types, represented by a set of triples of the form:

str(Index, Prosodic form, Type)

CatLog2 computes, for a given expression of English and Type, its associated logical forms according to the calculus of f9.1.1.pl and the lexicon. The lexicon satisfies the unit and bracket non-negativity conditions.

CatLog2 runs under XGP Prolog version 1.1.1 (19) which uses GNU Prolog. Within Prolog, once f9.1.1.pl is consulted, a query

?- pplex.

pretty prints the lexicon in the Prolog console. In the present case this is as follows:

```
GNU Prolog - version 1.3.0 of Nov 21 2006.
Copyright (C) 1999-2006 Daniel Diaz

compiling /Users/glynmorrill/Glyn/CatLog/f9.1.1.pl for byte code...
/Users/glynmorrill/Glyn/CatLog/f9.1.1.pl compiled, 5916 lines read - 1116241 bytes written, 556 ms
007: iLAgNt(s(g)): 007
a: iLAg(Af(SfciiNt(s(g)))inSf)/CNs(g))): LALBEC[(A C) & (B C)]
admire: L((<>(EaNa-EgNt(s(g)))\Sf)/EaNa): ^LALB(Pres ((vadmire A) B))
And: iLAf(Sf/Sf): LAA
and: iLAf((?iLSf\[]-1[]-1Sf)/iLSf): [mapPhin,0,and]
and: iLAaAf(?iL(<>Na\Sf)\[]-1[]-1(<>Na\Sf))/iL(<>Na\Sf)): [mapPhin,[app,s,0],and]
and: iLAaAf(?iL(Sf/!Na)\[]-1[]-1(Sf/!Na))/iL(Sf/!Na)): [mapPhin,[app,s,0],and]
and: iLAf(?iL(Sf£/EaNa)\[]-1[]-1(Sf£/EaNa))/iL(Sf/EaNa)): [mapPhin,[app,s,0],and]
and: iLAwAaAbAf(iL((Sfci((((<>Na\Sf)o-Wv)/Nb))licWw)\[]-1[]-1((Sfci((((<>Na\Sf)o-Wv)/Nb))licWw))/^^iL((Sfci((((<>Na\Sf)o-Wv)/Nb))licWw)): LALBLC[(B C) & (A C)]
and: iLAfAa(?iL(((<>Na\Sf)/EbNb)\(<>Na\Sf))/EbNb)\[]-1[]-1(((<>Na\Sf)/EbNb)\(<>Na\Sf))/EbNb)): [mapPhin,[app,s,[app,s,0]],and]
and: iLAfAa(?iL(((<>Na\Sf)/EbNb)\(<>Na\Sf))/EbNb)\[]-1[]-1(((<>Na\Sf)/EbNb)\(<>Na\Sf))/iL(((<>Na\Sf)/EbNb)\(<>Na\Sf))/iL(((<>Na\Sf)/EbNb)\(<>Na\Sf))): [mapPhin,[app,s,[app,s,0]],and]
and: iLAfAa(iL(((<>Na\Sf)/EbNb+Eg((CNg/CNg)|+|(CNg/CNg)))\(<>Na\Sf))\[]-1[]-1(((<>Na\Sf)/EbNb+Eg((CNg/CNg)|+|(CNg/CNg)))\(<>Na\Sf))/iL(((<>Na\Sf)/EbNb+Eg((CNg/CNg)|+|(CNg/CNg)))\(<>Na\Sf))): LALBLCLD[(B C) D) & ((A C) D)]
and: iLAaAbAf(?iL(((<>Na\Sf)/(EcNc+CPb))\(<>Na\Sf))\[]-1[]-1(((<>Na\Sf)/(EcNc+CPb))\(<>Na\Sf))/iL(((<>Na\Sf)/(EcNc+CPb))\(<>Na\Sf))): [mapPhin,[app,s,[app,s,0]],and]
and: iLAaAbAf(?iL(((<>Na\Sf)/PPb)\[]-1[]-1(((<>Na\Sf)/PPb))/iL(((<>Na\Sf)/PPb)): [mapPhin,[app,s,[app,s,0]],and]
and: iLAaAbAf(?iL(((<>Na\Sf)/(EcNc*PPb))\(<>Na\Sf))\[]-1[]-1(((<>Na\Sf)/(EcNc*PPb))\(<>Na\Sf))/iL(((<>Na\Sf)/(EcNc*PPb))\(<>Na\Sf))): [mapPhin,[app,s,[app,s,0]],and]
and: iLAaAbAf(?iL(((<>Na\Sf)/Nb)\[]-1[]-1(((<>Na\Sf)/Nb))/iL(((<>Na\Sf)/Nb)): [mapPhin,[app,s,[app,s,0]],and]
ate: L((<>EaNa\Sf)/EaNa): ^LALB(Past ((veat A) B))
bagels: LQNt(p(n))&CNp(n)): ^(gen vbagels), vbagels)
barn: LCNs(n): barn
be: L(<>W[there]-oSb)/EaNa): ^LA(vbe A)
before: iL(AaAf(<>Na\Sf)\(<>Na\Sf))/Sf): LALBLC((before A) (B C))
beginning: LCNs(n): beginning
believes: L((<>EgNt(s(g))\Sf)/(CPthat|+|LSf)): ^LALB(Pres ((vbelieve A) B))
bill: iLNt(s(m)): b
bond: iLNt(s(m)): b
book: LCNs(n): book
bought: L((<>EaNa\Sf)/(EaNa*EaNa)): ^LALB(Past (((vbuy piiA) pi2A) B))
bought: L((<>EaNa\Sf)/EaNa): ^LALB(Past ((vbuy A) B))
by: iLAa(((<>Na\S_)\(<>Na\S_))/Na): LALBLC[[C = A] & (B C)]
by: L(An(CNn\CNn)/EaNa): ^LALB((vby A) B)
buys: L((<>EgNt(s(g))\Sf)/(EaNa*EaNa)): ^LALB(Pres (((vbuy piiA) pi2A) B))
calls: L((<>EgNt(s(g))\Sf)/Ea(CWlup].oNa)|+|CNao.W[up])): ^LALB(vphone A) B)
catch: L((<>EaNa\Sb)/EaNa): ^LALB((vcatch A) B)
cezanne: iLNt(s(m)): c
cd: LCNs(n): cd
charles: iLNt(s(m)): c
clark: iLAgNt(s(g)): c
coffee: LQt(s(n))&CNs(n)): ^(gen vcoffee), vcoffee)
created: L((<>EaNa\Sf)/EaNa): ^LALB(Past ((vcreate A) B))
darkness: L(CNs(n)&Nt(s(n))): ^(vdarkness, (gen vdarkness))
deep: LCNs(n): deep
did: iLAaAgAbAh(((<>Na\Sg)ci(<>Nb\Sh))/(Ec<>Nc\Sf))\((<>Na\Sg)ci(<>Nb\Sh))): LALB((A B) B)
did+too: (((<>NA\SB)ci(<>NC\SD))/(<>NE\SF))\(((<>NG\SH)ci(NI\SJ))): LKLL((K L) L)
doesnt: iLAgAa(CSgci((<>Na\Sf)/(<>Na\Sb)))inSp): LA-(A LBLC(B C))
dog: LCNs(n): dog
donuts: LQt(p(n))&CNp(n)): ^(gen vdonuts), vdonuts)
earth: LCNs(n): earth
eat: L((<>EaNa\Sb)/EaNa): ^LALB((veat A) B)
edinburgh: iLNt(s(n)): e
editor: L(AgCNs(g)/PPof): editor
every: iLAg(Af(SfciNt(s(g)))inSf)/CNs(g)): LALBAC[(A C) -> (B C)]
everyone: LAf((SfciAgNt(g))inSf): ^LAAB[(vperson B) -> (A B)]
face: LCNs(n): face
fell: L(Ea<>Na\Sf): ^LA(Past (vfall A))
filed: L((<>EgNt(s(g))\Sf)/EaNa): ^LALB(Past ((vfile A) B))
finds: L((<>EgNt(s(g))\Sf)/EaNa): ^LALB(Pres ((vfind A) B))
fish: LCNs(n): fish
for: iL(PPfor/EaNa): LAA
form: L(CNs(n)&Nt(s(n))): ^(vform, (gen vform))
fortunately: LAf(vSfinSf): fortunately
friends: L(CNp/PPof): friends
from: L((AaAf(<>Na\Sf)\(<>Na\Sf))&An(CNn\CNn))/EbNb): ^LA((vfromadv A), (vfromadn A))
gave: L((<>EaNa\Sf)/(EbNb*PPto)): ^LALB(Past (((vgive pi2A) pi1A) B))
gave: L((<>EgNt(s(g))\Sf)/(EaNao.W[the,cold,shoulder])): ^LALB(Past ((vshun A) B))
```

```
gave: L(((<>EaNa\Sf)/EaNa)/EaNa): ^LALBLC(Past (((vgive A) B) C))
girl: LCNs(f): girl
gives: L((<>EgNt(s(g))\Sf)/(EaNao.W[the,cold,shoulder])): ^LALB(Pres ((vshun A) B))
God: iLNt(s(m)): God
good: LAn(CNn/CNn): good
has: L((<>EgNt(s(g))\Sf)/EaNa): ^LALB(Pres ((vhave A) B))
he: iL[]-1Ag((iLSg lca iLNt(s(m)))/(<>Nt(s(m))\Sg)): LAA
heaven: LCNs(n): heaven
her: iLAgAa(((<>Na\Sg)ciiLNt(s(f)))in(iL(<>Na\Sg) lca iLNt(s(f)))): LAA
himself: iLAf(((<>Nt(s(m))\Sf)ciNt(s(m)))in(<>Nt(s(m))\Sf)): LALB((A B) B)
horse: LCNs(n): horse
in: L(AaAf((<>Na\Sf)\(<>Na\Sf))/EaNa): ^LALBLC((vin A) (B C))
in: L(Af(SfndSf)/EaNa): in
is: iL((<>EgNt(s(g))\Sf)/(EaNa+(Eg((CNg/CNg)|+|(CNg\CNg))-I))): LALB(Pres (A->C.[B = C]; D.((D LE[E = B]) B)))
it: iLW[it]: 0
it: iLAfAa(((<>Na\Sf)ciiLNt(s(n)))in(iL(<>Na\Sf) lca iLNt(s(n)))): LAA
it: iL[]-1Af((iLSf lca iLNt(s(n)))/(<>Nt(s(n))\Sf)): LAA
jogs: L(<>EgNt(s(g))\Sf): ^LA(Pres (vjog A))
john: iLNt(s(m)): j
laughs: L(<>EgNt(s(g))\Sf): ^LA(Pres (vlaugh A))
left: L(<>EgNt(s(g))\Sf): ^LA(Pres (vleave A))
let: L(Sim/Sb): let
light: L(CNs(n)&Nt(s(n))): ^(vlight, (gen vlight))
likes: L((<>EgNt(s(g))\Sf)/EaNa): ^LALB(Pres ((vlike A) B))
logic: L(Nt(s(n))&CNs(n)): ^((gen vlogic), vlogic)
london: iLNt(s(n)): l
loses: L((<>EgNt(s(g))\Sf)/EaNa): ^LALB(Pres ((vlose A) B))
love: L((<>EaNa\Sb)/EaNa): ^LALB((vlove A) B)
loved: LAaAb(((<>Na\S-)ciNb)dp(((<>Na\S-)ciNb)inAg(CNg\CNg))): ^(vlove, LALBLC[(B C) & ED((A C) D)])
loves: L((<>EgNt(s(g))\Sf)/EaNa): ^LALB(Pres ((vlove A) B))
man: LCNs(m): man
mary: iLNt(s(f)): m
met: L((<>EaNa\Sf)/EaNa): ^LALB(Past ((vmeet A) B))
more: iLAhAgAf((Sfci(((ShciNt(p(g)))inSh)/CNp(g)))in(Sf/^(CPthanciiL(((ShciNt(p(g)))inSh)/CNp(g))))): LALB[|LC(A LDLE[(D C) & (E C)])| > |LFv(B LGLH[(G F) & (H F)])|]
mountain: LCNs(n): mountain
moved: L(<>EaNa\Sf): ^LA(Past (vmove A))
necessarily: iL(SA/LSA): Nec
of: L((An(CNn\CNn)/iLEbNb)&(PPof/EaNa)): ^(vof, LAA)
or: iLAf((?iLSf\[]-1[]-1Sf)/iLSf): [mapPhin,0,or]
or: iLAaAf((?iL(<>Na\Sf)\[]-1[]-1(<>Na\Sf))/iL(<>Na\Sf)): [mapPhin,[app,s,0],or]
or: iLAf((?iL(Sf/(<>EgNt(s(g))\Sf))\[]-1[]-1(Sf/(<>EgNt(s(g))\Sf)))/iL(Sf/(<>EgNt(s(g))\Sf))): [mapPhin,[app,s,0],or]
or: iLAaAf((?iL(((<>Na\Sf)/EbNb)/EbNb)\[]-1[]-1(((<>Na\Sf)/EbNb)/EbNb))/iL(((<>Na\Sf)/EbNb)/EbNb)): [mapPhin,[app,s,[app,s,[app,s,0]]],or]
painting: L(CNs(n)/PPof): ^LA((vof A) vpainting)
paper: LCNs(n): paper
park: LCNs(n): park
past: LAaAf(((<>Na\Sf)\(<>Na\Sf))/EbNb): ^LALBLC((vpast A) (B C))
perseverance: L(Nt(s(n))&CNs(n)): ^((gen vperseverance), vperseverance)
peter: iLNt(s(m)): p
phonetics: L(Nt(s(n))&CNs(n)): ^((gen vphonetics), vphonetics)
praises: L((<>EgNt(s(g))\Sf)/EaNa): ^LALB(Pres ((vpraise A) B))
raced: L(<>EaNa\Sf): ^LA(Past (vrace A))
raced: LAaAb(((<>Na\S-)ciNb)dp(((<>Na\S-)ciNb)inAg(CNg\CNg))): ^(vrace2, LALBLC[(B C) & ED((A C) D)])
rains: L(<>W[it]-oSf): ^(Pres vitrains)
reading: L((<>EaNa\Spsp)/EaNa): ^LALB((vread A) B)
robin: iLAgNt(s(g)): r
said: L((<>EaNa\Sf)/Sim): ^LALB(Past ((vsay A) B))
saw: L((<>EaNa\Sf)/(EaNa+CPthat)): ^LALB(Past ((A->C.(vseee C); D.(vseet D)) B))
seeks: L((<>EgNt(s(g))\Sf)/LAaAf(((Na\Sf)/EbNb)\(Na\Sf))): ^LALB((vtries ^((vA vfind) B)) B)
sees: L((<>EgNt(s(g))\Sf)/EaNa): ^LALB(Pres ((vsee A) B))
sent: L((<>EaNa\Sf)/(EbNb*PPto)): ^LALB(Past (((vsent pi2A) pi1A) B))
sent: L(((<>EaNa\Sf)/EaNa)/EaNa): ^LALBLC(Past (((vsend A) B) C))
she: iL[]-1Ag((iLSg lca iLNt(s(f)))/(<>Nt(s(f))\Sg)): LAA
sings: L(<>EgNt(s(g))\Sf): ^LA(Pres (vsing A))
slept: L(<>EgNt(s(g))\Sf): ^LA(Past (vsleep A))
slowly: LAaAf(L(<>Na\Sf)\(<>LNa\Sf)): ^LALB(vslowly ^(vA vB))
sneezed: L(<>EgNt(s(g))\Sf): ^LA(Past (vsneeze A))
sold: L((<>EaNa\Sf)/(EbNb*PPfor)): ^LALB(Past (((vsell pi2A) pi1A) B))
someone: LAf((SfciiLAgNt(g))inSf): ^LAEB[(vperson B) & (A B)]
Spirit: LCNs(m): Spirit
studies: L((<>EgNt(s(g))\Sf)/EaNa): ^LALB(Pres ((vstudy A) B))
```

```
such+that: iLAn((CNn\CNn)/(Sf lca iLNt(n))): LALBLC[(B C) & (A C)]
suzy: iLNt(s(f)): s
talks: L(<>EgNt(s(g))\Sf): ^LA(Pres (vtalk A))
tall: LAg(CNg/CNg): tall
teetotal: LAn(CNn/CNn): ^LALB[(A B) & (vteetotal B)]
tenmilliondollars: LNt(s(n)): tenmilliondollars
than: iL(CPthan/LSf): LAA
that: iL(CPthat/LSf): LAA
that: iLAn(]-1[]-1(CNn\CNn)/iL((<>Nt(n)|&||iLNt(n))\Sf)): LALBLC[(B C) & (A C)]
the: iLAn(Nt(n)/CNn): iota
the+cold+shoulder: iLW[the,cold,shoulder]: 0
there: iLW[there]: 0
thinks: L((<>EgNt(s(g))\Sf)/(CPthat|+|LSf)): ^LALB(Pres ((vthink A) B))
to: iL((PPto/EaNa)|&|An((<>Nn\Si)/(<>Nn\Sb))): LAA
today: LAaAf((<>Na\Sf)\(<>Na\Sf)): ^LALB(vtoday (A B))
tries: L((<>EgNt(s(g))\Sf)/L(<>EgNt(s(g))\Si)): ^LALB((vtries ^(vA B)) B)
unicorn: LCNs(n): unicorn
up: iLW[up]: 0
upon: L((AbAf(<>Nb\Sf)\(<>Nb\Sf))&Ag(CNg\CNg))/EaNa): ^LA((vuponadv A), (vuponadn A))
void: LAg(CNg/CNg): void
walk: L(<>(EaNa-EgNt(s(g)))\Sf): ^LA(Pres (vwalk A))
walk: L(<>EaNa\Sb): ^LA(vwalk A)
walks: L(<>EgNt(s(g))\Sf): ^LA(Pres (vwalk A))
was: iL((<>EgNt(s(g))\Sf)/(EaNa+(Eg((CNg/CNg)+|(CNg\CNg)-I))): LALB(Past (A->C.[B = C]; D.(CD LE[E = B]) B)))
was: L((<>W[there]-oSf)/EaNa): ^LA(Past (vbe A))
waters: LCNp(n): waters
which: iLAnAm(Nt(n)ciNt(m))in(]-1[]-1(CNm\CNm)/iL((<>Nt(n)|&||iLNt(n))\Sf))): LALBLCLD[(C D) & (B (A D))]
who: iLAhAm(]-1[]-1Nt(n)\((SheiNt(n))inSh))/iL((<>Nt(m)|&||iLNt(m))\Sf)): LALBLC[(A B) & (C B)]
will: iLAa((<>Na\Sf)/(<>Na\Sb)): LALB(Fut (A B))
without: L(Ag(CNg\CNg)/EaNa): ^LALBLC[(B C) & -((vwith A) C)]
without: iLAaAf(]-1((<>Na\Sf)\(<>Na\Sf))/(<>Na\Spsp)): LALBLC[(B C) & -(A C)]
woman: LCNs(f): woman
yesterday: LAaAf((<>Na\Sf)\(<>Na\Sf)): ^LALB(vyesterday (A B))
```

Part I

Categorial Grammar and Listout of CatLog2

Chapter 2

Logical translations and their proofs

A query

?- t(*N*).

parses all the strings the index of which unifies with the Prolog term *N*. For example, a session with ?- t(1) is as follows, with f9.1.1.pl as in Chapter 3 already consulted.

```
(1) [john]+walks Sf


[iLNt(s(m)): j], L(<>EgNt(s(g))\Sf): ^LA(Pres (vwalk A)) => Sf


[iLNt(s(m))], L(<>EgNt(s(g))\Sf) => Sf [LL]
    [iLNt(s(m))], <>EgNt(s(g))\Sf => Sf [\L]
        [iLNt(s(m))] => <>EgNt(s(g)) [<>R]
            iLNt(s(m)) => EgNt(s(g)) [ER]
                iLNt(s(m)) => Nt(s(m)) [iLL]
                    Nt(s(m)) => Nt(s(m))
        Sf => Sf

(Pres (vwalk j))
```

We see first the prosodic form (bracketed string) and the type in which it is to be analysed. Then there is the associated logic statement a proof of which would show grammaticality, including the lexical semantics of the basic expressions. Then there is such a proof. Finally there is the logical translation thus results from all this.

Chapter 3

Listout of the program CatLog2 (CatLog version f9.1.1)

```
/*

This is CatLog2 - CatLog version f9.1.1. of May 25 2018.
Copyright (C) Glyn V. Morrill

This is the categorial parser/theorem-prover CatLog2 developed
by Glyn Morrill at the Departament of Computer Science,
Polytecnic University of Catalunya.
It is made available for non-comercial research purposes only,
and provided the authorship is acknowledged. There is absolutely
no warranty. All rights are reserved.

Once consulted, pplex prints the lexicon
within Prolog and pplexlatex pretty prints the
lexicon in LaTeX to file s.tex. t(M) tests examples
unifying with M and writes analyses within Prolog
and an analysis for each reading in LaTeX to file t.tex,
which requires Paul Taylor's prooftree.sty to be processed
by LaTeX.

25th May 2018

*/

% atfoc(inp) or atfoc(out)

atfoc(inp).

:- op(450, xfy, :).
:- op(500, xfx, '=>').

:- op(400, xfx, /).    % 1    over
:- op(400, xfx, bs).   % 2    under
:- op(400, xfx, *).    % 3    continuous product
% i                    % 4    continuous unit

:- op(400, xfx, ci).   % 5+   circumfix (leftmost)
:- op(400, xfx, cin).  % 5-   circumfix (rightmost)
:- op(400, xfx, in).   % 6+   infix (leftmost)
:- op(400, xfx, inn).  % 6-   infix (rightmost)
:- op(400, xfx, dp).   % 7+   discontinuous product (leftmost)
:- op(400, xfx, dpn).  % 7-   discontinuous product (rightmost)
% j                    % 8    discontinuous unit

:- op(400, xfx, &).    % 9    additive conjunction
:- op(400, xfy, +).    % 10   additive disjunction

:- op(300, xfy, u).    % 11   1st order univ. qu.
:- op(300, xfy, e).    % 12   1st order exist. qu.

:- op(300, fy, 'L').   % 13   universal modality
:- op(300, fy, 'M').   % 14   existential modality

:- op(300, fy, ab).    % 15   exist. bracket modality
:- op(300, fy, br).    % 16   univ. bracket modality

:- op(300, fy, '!').   % 17   universal exponential
:- op(300, fy, '?').   % 18   existential exponential

:- op(400, xfx, lca).  % 19   contr. for anaphora

:- op(400, xfx, lio).  % 20   left sem. inactive over
:- op(400, xfx, liu).  % 21   left sem. inactive under
:- op(400, xfx, rio).  % 22   right sem. inactive over
:- op(400, xfx, riu).  % 23   right sem. inactive under
:- op(400, xfx, lip).  % 24   left sem. inactive cont. product
:- op(400, xfx, rip).  % 25   right sem. inactive cont. product
```

```
:- op(400, xfx, uic).  % 26+   upper sem. inactive circumfix (leftmost)
:- op(400, xfx, uicn). % 26-   upper sem. inactive circumfix (rightmost)
:- op(400, xfx, uii).  % 27+   upper sem. inactive infix (leftmost)
:- op(400, xfx, uiin). % 27-   upper sem. inactive infix (rightmost)
:- op(400, xfx, lic).  % 28+   lower sem. inactive circumfix (leftmost)
:- op(400, xfx, licn). % 28-   lower sem. inactive circumfix (rightmost)
:- op(400, xfx, lii).  % 29+   lower sem. inactive infix (leftmost)
:- op(400, xfx, liin). % 29-   lower sem. inactive infix (rightmost)
:- op(400, xfx, uidp). % 30+   upper sem. inactive disc. product (leftmost)
:- op(400, xfx, uidpn).% 30-   upper sem. inactive disc. product (rightmost)
:- op(400, xfx, lidp). % 31+   lower sem. inactive disc. product (leftmost)
:- op(400, xfx, lidpn).% 31-   lower sem. inactive disc. product (rightmost)

% w(w)                 % 32    words as types

:- op(400, xfx, iac).  % 33    sem. inactive additive conjunction
:- op(400, xfx, iad).  % 34    sem. inactive additive disjunction

:- op(300, xfy, iu).   % 35    sem. inactive 1st order univ. qu.
:- op(300, xfy, ie).   % 36    sem. inactive 1st order exist. qu.

:- op(300, fy, iL).    % 37    sem. inactive universal modality
:- op(300, fy, iM).    % 38    sem. inactive existential modality

:- op(300, fy, lp).    % 39    left projection
:- op(300, fy, rp).    % 40    right projection
:- op(300, fy, li).    % 41    left injection
:- op(300, fy, ri).    % 42    right projection

:- op(300, fy, sp).    % 43+   split (leftmost)
:- op(300, fy, spn).   % 43-   split (rightmost)
:- op(300, fy, bg).    % 44+   bridge (leftmost)
:- op(300, fy, bgn).   % 44-   bridge (rightmost)

:- op(400, xfx, nd).   % 45    nondet. division
:- op(400, xfx, np).   % 46    nondet. continuous product

:- op(400, xfx, nci).  % 47    nondet. circumfix
:- op(400, xfx, nin).  % 48    nondet. infix
:- op(400, xfx, ndp).  % 49    nondet. discontinuous product

:- op(400, xfx, -).    % 50    difference

doout(Gamma, A, S) :-
      nl, nl,  ppseq(user, yes, []: Gamma, l(A)), nl,
      nl(S), nl(S), write(S, `\vspace{0.15in}`),
      nl(S), write(S, '$'), ppseq(latex(S), yes, []: Gamma, l(A)),
      write(S, '$'), nl(S), (!).

prs(N, S) :-
      fetch(N, Str, A, S),
      lookup(Str, Gamma),
      doout(Gamma, A, S),
      export(S),
      clean,
      p([]: Gamma, l(A, Phi), Prf),
      nl, nl, pprf(user, no, 0, Prf),
      eval(Phi, NF),
      nl, nl, ppsem(user, NF), nl,
      assert(found(Prf, NF)),
      fail.

prs(_, S) :- close(S).

t(N) :- %clean,
      open('t.tex', write, S),
      prs(N, S).

clean :- retract(copy(_)), (!),
      clean.
```

```
clean :- retract(copy(_,_,_)), (!),
        clean.

clean :- retract(found(_, _)), (!),
        clean.

%clean :- retract(counter(_)), (!),
%     clean.

clean :- retract(symb(_)), (!),
        clean.

clean :- %assert(counter(0)),
     assert(symb(0)).

gensymb(S1) :- retract(symb(S)),
        S1 is S+1,
        assert(symb(S1)).

allumodszone([]: Gamma) :-
     allumodsconfig(Gamma).

allumodsconfig([H|T]) :- var(H), (!),
     allumodsconfig(T).

allumodsconfig([1|T]) :-
     allumodsconfig(T).

allumodsconfig([]).

allumodsconfig([1('L'_,_,_, Ls)|Gamma]) :-
     allumodsconfiglst(Ls),
     allumodsconfig(Gamma).

allumodsconfig([1(iL _,_, Ls)|Gamma]) :-
     allumodsconfiglst(Ls),
     allumodsconfig(Gamma).

allumodsconfig([b(OMEGA)|Gamma]) :-
     allumodszone(OMEGA),
     allumodsconfig(Gamma).

allumodsconfiglst([]).

allumodsconfiglst([H|T]) :-
     allumodsconfig(H),
     allumodsconfiglst(T).

fetch(N, Str, A, S) :-
     str(N, Str, A),
     nl, nl, write('('), write(N), write(') '),
     nl(S), nl(S), write(S, '\vspace{0.15in}'), nl(S),
     write(S, ' ( '), write(S, N), write(S, ' ) '),
     ppprs(user, Str), write(' '), pptype(user, A), nl,
     write(S, '$'), ppprs(latex(S), Str), write(S, '$'), nl(S).

exportone(S) :- collectall([], Derivs),
     exportall(Derivs, S).

exportone(S) :- close(S).

export(_).

export(S) :- collectall([], Derivs),
     exportall(Derivs, S).

collectall(InDerivs, OutDerivs) :-
     retract(found(Prf, NF)), (!),
```

```
     numbervars(NF, 0, _),
     addifnew(Prf, NF, InDerivs, Derivs),
     collectall(Derivs, OutDerivs).

collectall(Derivs, Derivs).

addifnew(_, NF, Derivs, Derivs) :-
      member(f(_, NF), Derivs), (!).

addifnew(Prf, NF, Derivs, [f(Prf, NF)|Derivs]).

% exportall([], _).

exportall([f(Prf, Phi)|Derivs], S) :-
      nl(S), nl(S), write(S, `\vspace{0.15in}`), nl(S), nl(S),
      ppprf(latex(S), no, _, Prf),
      nl(S), nl(S), write(S, `\vspace{0.15in}`), nl(S), nl(S),
      write(S, '$'), ppsem(latex(S), Phi), write(S, '$'),
      exportall(Derivs, S).

p(OMEGA, l(A, Phi), Prf) :-
%      countcheck(OMEGA, A),
      p1(OMEGA, l(A, Phi), Prf).

primitive(n(_), []).
primitive(s(_), []).
primitive(pp(_), []).
primitive(cp(_), []).
primitive(cn(_), []).

primitive(si, [1]).

primitive(q, []).

primitive(w(_), []).

% countcheck(+OMEGA, +A) checks that the sequent OMEGA => A
% is not unprovable by reason of the count invariant.

countcheck(OMEGA, A) :-
      \+((typecount(max, out, A, V1),
         zonecount(min, OMEGA, V2),
         listminus(V2, V1, V),
         \+allgezero(V))),
      \+((typecount(min, out, A, V3),
         zonecount(max, OMEGA, V4),
         listminus(V4, V3, VV),
         \+alllezero(VV))).

zonecount(M, []: Gamma, V) :-
      configcount(M, Gamma, V).

allgezero([]).

allgezero([C|V]) :- C >= 0,
      allgezero(V).

alllezero([]).

alllezero([C|V]) :- C =< 0,
      alllezero(V).

comp(inp, out).
comp(out, inp).

comp(min, max).
comp(max, min).

typecount(_, _, n(_), [1, 0, 0, 0, 0, 0, 0]).
typecount(_, _, s(_), [0, 1, 0, 0, 0, 0, 0]).
```

```
typecount(_, _, pp(_),[0, 0, 1, 0, 0, 0, 0, 0]).
typecount(_, _, cp(_),[0, 0, 0, 1, 0, 0, 0, 0]).
typecount(_, _, cn(_),[0, 0, 0, 0, 1, 0, 0, 0]).

typecount(_, _, si,   [0, 0, 0, 0, 0, 0, 1, 0]).

typecount(_, _, q,    [0, 0, 0, 0, 0, 0, 0, 1]).

typecount(M, Pol, C/B, V) :-                                    % 1
      typecount(M, Pol, C, V1),
      comp(Pol, Poll), comp(M, M1),
      typecount(M1, Poll, B, V2),
      listminus(V1, V2, V).

typecount(M, Pol, A bs C, V) :-                                 % 2
      typecount(M, Pol, C, V1),
      comp(Pol, Poll), comp(M, M1),
      typecount(M1, Poll, A, V2),
      listminus(V1, V2, V).

typecount(M, Pol, A*B, V) :-                                    % 3
      typecount(M, Pol, A, V1),
      typecount(M, Pol, B, V2),
      listplus(V1, V2, V).

typecount(_, _, i, [0, 0, 0, 0, 0, 0, 0, 0]).                   % 4

typecount(M, Pol, C ci B, V) :-                                 % 5+
      typecount(M, Pol, C, V1),
      comp(Pol, Poll), comp(M, M1),
      typecount(M1, Poll, B, V2),
      listminus(V1, V2, V).

typecount(M, Pol, C cin B, V) :-                                % 5-
      typecount(M, Pol, C, V1),
      comp(Pol, Poll), comp(M, M1),
      typecount(M1, Poll, B, V2),
      listminus(V1, V2, V).

typecount(M, Pol, A in C, V) :-                                 % 6+
      typecount(M, Pol, C, V1),
      comp(Pol, Poll), comp(M, M1),
      typecount(M1, Poll, A, V2),
      listminus(V1, V2, V).

typecount(M, Pol, A inn C, V) :-                                % 6-
      typecount(M, Pol, C, V1),
      comp(Pol, Poll), comp(M, M1),
      typecount(M1, Poll, A, V2),
      listminus(V1, V2, V).

typecount(M, Pol, A dp B, V) :-                                 % 7+
      typecount(M, Pol, A, V1),
      typecount(M, Pol, B, V2),
      listplus(V1, V2, V).

typecount(M, Pol, A dpn B, V) :-                                % 7-
      typecount(M, Pol, A, V1),
      typecount(M, Pol, B, V2),
      listplus(V1, V2, V).

typecount(_, _, j, [0, 0, 0, 0, 0, 0, 0, 0]).                   % 8

typecount(M, inp, A&B, V) :-                                    % 9
      typecount(M, inp, A, V1),
      typecount(M, inp, B, V2), comp(M, M1),
      listm(M1, V1, V2, V).

typecount(M, out, A&B, V) :-                                    % 9
      typecount(M, out, A, V1),
```

```
      typecount(M, out, B, V2),
      listm(M, V1, V2, V).

typecount(M, inp, A+B, V) :-                    % 10
      typecount(M, inp, A, V1),
      typecount(M, inp, B, V2),
      listm(M, V1, V2, V).

typecount(M, out, A+B, V) :-                    % 10
      typecount(M, out, A, V1),
      typecount(M, out, B, V2), comp(M, M1),
      listm(M1, V1, V2, V).

typecount(M, Pol, _ u A, V) :-                  % 11
      typecount(M, Pol, A, V).

typecount(M, Pol, _ e A, V) :-                  % 12
      typecount(M, Pol, A, V).

typecount(M, Pol, 'L'A, V) :-                   % 13
      typecount(M, Pol, A, V).

typecount(M, Pol, 'M'A, V) :-                   % 14
      typecount(M, Pol, A, V).

typecount(M, Pol, ab A, V) :-                   % 15
      typecount(M, Pol, A, V).

typecount(M, Pol, br A, V) :-                   % 16
      typecount(M, Pol, A, V).

% sharing: 17, 18, 19

typecount(M, Pol, C lio B, V) :-                % 20
      typecount(M, Pol, C, V1),
      comp(Pol, Pol1), comp(M, M1),
      typecount(M1, Pol1, B, V2),
      listminus(V1, V2, V).

typecount(M, Pol, A liu C, V) :-                % 21
      typecount(M, Pol, C, V1),
      comp(Pol, Pol1), comp(M, M1),
      typecount(M1, Pol1, A, V2),
      listminus(V1, V2, V).

typecount(M, Pol, C rio B, V) :-                % 22
      typecount(M, Pol, C, V1),
      comp(Pol, Pol1), comp(M, M1),
      typecount(M1, Pol1, B, V2),
      listminus(V1, V2, V).

typecount(M, Pol, A riu C, V) :-                % 23
      typecount(M, Pol, C, V1),
      comp(Pol, Pol1), comp(M, M1),
      typecount(M1, Pol1, A, V2),
      listminus(V1, V2, V).

typecount(M, Pol, A lip B, V) :-                % 24
      typecount(M, Pol, A, V1),
      typecount(M, Pol, B, V2),
      listplus(V1, V2, V).

typecount(M, Pol, A rip B, V) :-                % 25
      typecount(M, Pol, A, V1),
      typecount(M, Pol, B, V2),
      listplus(V1, V2, V).

typecount(M, Pol, C uic B, V) :-                % 26+
      typecount(M, Pol, C, V1),
      comp(Pol, Pol1), comp(M, M1),
```

```
    typecount(M1, Poll, B, V2),
    listminus(V1, V2, V).

typecount(M, Pol, C uicn B, V) :-                    % 26-
    typecount(M, Pol, C, V1),
    comp(Pol, Poll), comp(M, M1),
    typecount(M1, Poll, B, V2),
    listminus(V1, V2, V).

typecount(M, Pol, A uii C, V) :-                     % 27+
    typecount(M, Pol, C, V1),
    comp(Pol, Poll), comp(M, M1),
    typecount(M1, Poll, A, V2),
    listminus(V1, V2, V).

typecount(M, Pol, A uin C, V) :-                     % 27-
    typecount(M, Pol, C, V1),
    comp(Pol, Poll), comp(M, M1),
    typecount(M1, Poll, A, V2),
    listminus(V1, V2, V).

typecount(M, Pol, C lic B, V) :-                     % 28+
    typecount(M, Pol, C, V1),
    comp(Pol, Poll), comp(M, M1),
    typecount(M1, Poll, B, V2),
    listminus(V1, V2, V).

typecount(M, Pol, C licn B, V) :-                    % 28-
    typecount(M, Pol, C, V1),
    comp(Pol, Poll), comp(M, M1),
    typecount(M1, Poll, B, V2),
    listminus(V1, V2, V).

typecount(M, Pol, A lii C, V) :-                     % 29+
    typecount(M, Pol, C, V1),
    comp(Pol, Poll), comp(M, M1),
    typecount(M1, Poll, A, V2),
    listminus(V1, V2, V).

typecount(M, Pol, A liin C, V) :-                    % 29-
    typecount(M, Pol, C, V1),
    comp(Pol, Poll), comp(M, M1),
    typecount(M1, Poll, A, V2),
    listminus(V1, V2, V).

typecount(M, Pol, A uidp B, V) :-                    % 30+
    typecount(M, Pol, A, V1),
    typecount(M, Pol, B, V2),
    listplus(V1, V2, V).

typecount(M, Pol, A uidpn B, V) :-                   % 30-
    typecount(M, Pol, A, V1),
    typecount(M, Pol, B, V2),
    listplus(V1, V2, V).

typecount(M, Pol, A lidp B, V) :-                    % 31+
    typecount(M, Pol, A, V1),
    typecount(M, Pol, B, V2),
    listplus(V1, V2, V).

typecount(M, Pol, A lidpn B, V) :-                   % 31-
    typecount(M, Pol, A, V1),
    typecount(M, Pol, B, V2),
    listplus(V1, V2, V).

typecount(_, _, w(_), [0, 0, 0, 0, 0, 0, 0, 0]).    % 32

typecount(M, inp, A iac B, V) :-                     % 33
    typecount(M, inp, A, V1),
    typecount(M, inp, B, V2), comp(M, M1),
```

```
      listm(M1, V1, V2, V).

typecount(M, out, A iac B, V) :-                  % 33
      typecount(M, out, A, V1),
      typecount(M, out, B, V2),
      listm(M, V1, V2, V).

typecount(M, inp, A iad B, V) :-                  % 34
      typecount(M, inp, A, V1),
      typecount(M, inp, B, V2),
      listm(M, V1, V2, V).

typecount(M, out, A iad B, V) :-                  % 34
      typecount(M, out, A, V1),
      typecount(M, out, B, V2), comp(M, M1),
      listm(M1, V1, V2, V).

typecount(M, Pol, _ iu A, V) :-                   % 35
      typecount(M, Pol, A, V).

typecount(M, Pol, _ ie A, V) :-                   % 36
      typecount(M, Pol, A, V).

typecount(M, Pol, iL A, V) :-                     % 37
      typecount(M, Pol, A, V).

typecount(M, Pol, iM A, V) :-                     % 38
      typecount(M, Pol, A, V).

typecount(M, Pol, lp A, V) :-                     % 39
      typecount(M, Pol, A, V).

typecount(M, Pol, rp A, V) :-                     % 40
      typecount(M, Pol, A, V).

typecount(M, Pol, li A, V) :-                     % 41
      typecount(M, Pol, A, V).

typecount(M, Pol, ri A, V) :-                     % 42
      typecount(M, Pol, A, V).

typecount(M, Pol, sp A, V) :-                     % 43+
      typecount(M, Pol, A, V).

typecount(M, Pol, spn A, V) :-                    % 43-
      typecount(M, Pol, A, V).

typecount(M, Pol, bg A, V) :-                     % 44+
      typecount(M, Pol, A, V).

typecount(M, Pol, bgn A, V) :-                    % 44-
      typecount(M, Pol, A, V).

typecount(M, Pol, C nd B, V) :-                   % 45
      typecount(M, Pol, C, V1),
      comp(Pol, Pol1), comp(M, M1),
      typecount(M1, Pol1, B, V2),
      listminus(V1, V2, V).

typecount(M, Pol, A np B, V) :-                   % 46
      typecount(M, Pol, A, V1),
      typecount(M, Pol, B, V2),
      listplus(V1, V2, V).

typecount(M, Pol, C nci B, V) :-                  % 47
      typecount(M, Pol, C, V1),
      comp(Pol, Pol1), comp(M, M1),
      typecount(M1, Pol1, B, V2),
      listminus(V1, V2, V).
```

```
typecount(M, Pol, A nin C, V) :-                    % 48
        typecount(M, Pol, C, V1),
        comp(Pol, Pol1), comp(M, M1),
        typecount(M1, Pol1, A, V2),
        listminus(V1, V2, V).

typecount(M, Pol, A ndp B, V) :-                    % 49
        typecount(M, Pol, A, V1),
        typecount(M, Pol, B, V2),
        listplus(V1, V2, V).

typecount(M, Pol, A - _, V) :-                      % 50
        typecount(M, Pol, A, V).

listm(_, [], [], []).

listm(M, [C1|V1], [C2|V2], [C|V]) :-
        m(M, C1, C2, C),
        listm(M, V1, V2, V).

m(min, C1, C2, C1) :- C1 =< C2.

m(min, C1, C2, C2) :- C2 < C1.

m(max, C1, C2, C1) :- C1 >= C2.

m(max, C1, C2, C2) :- C2 > C1.

configcount(_, [], [0, 0, 0, 0, 0, 0, 0, 0]).

configcount(M, [1|Gamma], V) :-
        configcount(M, Gamma, V).

configcount(M, [l(A, _, Ls)|Gamma], V) :-
        ltypecount(M, A, Ls, V1),
        configcount(M, Gamma, V2),
        listplus(V1, V2, V).

configcount(M, [f(A, _, Ls)|Gamma], V) :-
        ltypecount(M, A, Ls, V1),
        configcount(M, Gamma, V2),
        listplus(V1, V2, V).

configcount(M, [b([]: Gamma)|Gamma1], V) :-
        configcount(M, Gamma, V1),
        configcount(M, Gamma1, V2),
        listplus(V1, V2, V).

ltypecount(M, A, Ls, V) :-
        typecount(M, inp, A, V1),
        configlstcount(M, Ls, V2),
        listplus(V1, V2, V).

configlstcount(_, [], [0, 0, 0, 0, 0, 0, 0, 0]).

configlstcount(M, [L|Ls], V) :-
        configcount(M, L, V1),
        configlstcount(M, Ls, V2),
        listplus(V1, V2, V).

listminus([], [], []).

listminus([X|N], [Y|M], [Z|P]) :-
        Z is X-Y,
        listminus(N, M, P).

listplus([], [], []).

listplus([X|N], [Y|M], [Z|P]) :-
        Z is X+Y,
```

```
      listplus(N, M, P).

% pone(+Zone, ?SuccPhi, -Premises, -Prf) means that Zone => Succ is proved with semantics
% Phi with a reversible rule last if Premises can be proved, and the proof is Prf.

pone([]: [l(A, Phi, [])], l(A, Phi), [], prf(id, []: [l(A, Phi, [])], l(A), [])) :-
      primitive(A, []).

pone([]: [l(A, Phi, [[1]])], l(A, Phi), [], prf(id, []: [l(A, Phi, [[1]])], l(A), [])) :-
      primitive(A, [1]).


% 1

pone(Zeta: Gamma, l(C/B, [lmd, Y, Chi]), [Zeta: GammaB => l(C, Chi): SubPrf], prf(right(over), Zeta: Gamma, l(C/B), [SubPrf])) :-
      vector(B, Y, Bvec),
      append(Gamma, [Bvec], GammaB).

% 2

pone(Zeta: Gamma, l(A bs C, [lmd, X, Chi]), [Zeta: [Avec|Gamma] => l(C, Chi): SubPrf], prf(right(under), Zeta: Gamma, l(A bs C), [SubPrf])) :-
      vector(A, X, Avec).

% 3

pone(OMEGA, l(D, Omega), [OMEGAPr => l(D, Omega): SubPrf], prf(left(product), OMEGA, l(D), [SubPrf])) :-
      subconfigofzoneandapp(OMEGA, OMEGAPr, H, Deltal, [l(A*B, Chi, Ls)], Deltar),
      ssort(A, S),
      kappend(S, Ls1, Ls2, Ls),
      append(Deltal, [l(A, [fst, Chi], Ls1), l(B, [snd, Chi], Ls2)|Deltar], H).

% 4

pone(OMEGA, l(A, Phi), [OMEGAPr => l(A, Phi): SubPrf], prf(left(productunit), OMEGA, l(A), [SubPrf])) :-
      subconfigofzoneandapp(OMEGA, OMEGAPr, H, Deltal, [l(i, _, [])], Deltar),
      append(Deltal, Deltar, H).

% 5+

pone(Zeta: Gamma, l(C ci B, [lmd, Y, Chi]), [Zeta: GammaPr => l(C, Chi): SubPrf], prf(right(circumfix), Zeta: Gamma, l(C ci B), [SubPrf])) :-
      vector(B, Y, Bvec),
      configsort(Gamma, S),
      do_ones(S, [[1]|Ls]),
      fold(S, Gamma, [[Bvec]|Ls], GammaPr).

% 5-

pone(Zeta: Gamma, l(C cin B, [lmd, Y, Chi]), [Zeta: GammaPr => l(C, Chi): SubPrf], prf(right(circumfixn), Zeta: Gamma, l(C cin B), [SubPrf])) :-
      vector(B, Y, Bvec),
      configsort(Gamma, S),
      do_ones(S, [[1]|Ls]),
      append(Ls, [[Bvec]], LsB),
      fold(S, Gamma, LsB, GammaPr).

% 6+

pone(Zeta: Gamma, l(A in C, [lmd, X, Chi]), [Zeta: GammaPr => l(C, Chi): SubPrf], prf(right(infix), Zeta: Gamma, l(A in C), [SubPrf])) :-
      vector(A, X, Avec),
      ssort(A, S),
      do_ones(S, [[1]|Ls]),
      fold(S, Avec, [[Gamma]|Ls], GammaPr).

% 6-

pone(Zeta: Gamma, l(A inn C, [lmd, X, Chi]), [Zeta: GammaPr => l(C, Chi): SubPrf], prf(right(infixn), Zeta: Gamma, l(A in C), [SubPrf])) :-
      vector(A, X, Avec),
      ssort(A, S),
      do_ones(S, [[1]|Ls]),
      append(Ls, Gamma, LsGamma),
      fold(S, Avec, LsGamma, GammaPr).
```

```
% 7+

pone(OMEGA, l(D, Omega), [OMEGAPr => l(D, Omega): SubPrf], prf(left(dprod), OMEGA, l(D), [SubPrf])) :-
    subconfigofzoneandapp(OMEGA, OMEGAPr, H, Deltal, [l(A dp B, Chi, Ls)], Deltar),
    ssort(B, S),
    kappend(S, Ls1, Ls2, Ls),
    append(Deltal, [l(A, [fst, Chi], [[l(B, [snd, Chi], Ls1)]|Ls2])|Deltar], H).

% 7-

pone(OMEGA, l(D, Omega), [OMEGAPr => l(D, Omega): SubPrf], prf(left(dprodn), OMEGA, l(D), [SubPrf])) :-
    subconfigofzoneandapp(OMEGA, OMEGAPr, H, Deltal, [l(A dpn B, Chi, Ls)], Deltar),
    ssort(B, S),
    kappend(S, Ls1, Ls2, Ls),
    append(Ls2, [[l(B, [snd, Chi], Ls1)]], Ls2B),
    append(Deltal, [l(A, [fst, Chi], Ls2B)|Deltar], H).

% 8

pone(OMEGA, l(A, Phi), [OMEGAPr => l(A, Phi): SubPrf], prf(left(dproductunit), OMEGA, l(A), [SubPrf])) :-
    subconfigofzoneandapp(OMEGA, OMEGAPr, H, Deltal, [l(j, -, [Gamma])], Deltar),
    nappend([Deltal, Gamma, Deltar], H).

% 9

pone(OMEGA, l(A&B, [pair, Phi, Psi]), [OMEGA => l(A, Phi): SubPrf1, OMEGA => l(B, Psi): SubPrf2], prf(right(acon)), OMEGA, l(A&B), [SubPrf1, SubPrf2])).

% 10

pone(OMEGA, l(C, [case, Chi, X, Chi1, Y, Chi2]), [OMEGAPr1 => l(C, Chi1): SubPrf1, OMEGAPr2 => l(C, Chi2): SubPrf2], prf(left(adisj)), OMEGA, l(C), [SubPrf1, SubPrf2])) :-
    leafofsubconfigofzone(OMEGA, OMEGAPr, xxx, l(A+B, Chi, Ls)),
    leafofsubconfigofzone(OMEGAPr, OMEGAPr1, l(A, X, Ls), xxx),
    leafofsubconfigofzone(OMEGAPr, OMEGAPr2, l(B, Y, Ls), xxx).

% 11

pone(OMEGA, l(V u A, [lmd, V, Phi]), [OMEGA => l(A1, Phi): SubPrf], prf(right(univq), OMEGA, l(V u A), [SubPrf])) :-
    gensymb(S),
    tsubst(S, V, A, A1).

% 12

pone(OMEGA, l(B, Psi), [OMEGAPr => l(B, Psi): SubPrf], prf(left(exstq), OMEGA, l(B), [SubPrf])) :-
    leafofsubconfigofzone(OMEGA, OMEGAPr, l(A1, X, Ls), l(V e A, [snd, X], Ls)),
    gensymb(S),
    tsubst(S, V, A, A1).

% 13

pone(OMEGA, l('l'A, [up, Phi]), [OMEGA => l(A, Phi): SubPrf], prf(right(umod), OMEGA, l('l'A), [SubPrf])) :-
    allumodszone(OMEGA).

% 14

pone(OMEGA, l('M'B, Psi), [OMEGAPr => l('M'B, Psi): SubPrf], prf(left(emod), OMEGA, l('M'B), [SubPrf])) :-
    leafofsubconfigofzone(OMEGA, OMEGAPr, H, l('M'A, Chi, Ls)),
    allumodszone(OMEGAPr),
    H = l(A, [cup, Chi], Ls).

pone(OMEGA, l(iM B, Psi), [OMEGAPr => l('M'B, Psi): SubPrf], prf(left(emod), OMEGA, l('M'B), [SubPrf])) :-
    leafofsubconfigofzone(OMEGA, OMEGAPr, H, l('M'A, Chi, Ls)),
    allumodszone(OMEGAPr),
    H = l(A, [cup, Chi], Ls).

% 15

pone([]: Gamma, l(ab A, Phi), [[]: [[]: b(Gamma)] => l(A, Phi): SubPrf], prf(right(abrack), []: Gamma, l(ab A), [SubPrf])).

% 16
```

```
pone(OMEGA, l(B, Psi), [OMEGAPr => l(B, Psi): SubPrf], prf(left(brack), OMEGA, l(B), [SubPrf])) :-
      leafofsubconfigofzone(OMEGA, OMEGAPr, b([]: [l(A, Phi, Ls)]), l(br A, Phi, Ls)).

% 17

pone(Zeta: [], l(!A, Phi), [Zeta: [] => l(A, Phi): SubPrf], prf(right(uexp), Zeta: [], l(!A), [SubPrf])).

pone(Zeta: GammaAGamma, l(B, Psi), [[l(A, Phi, [])|Zeta]: GammaGamma => l(B, Psi): SubPrf], prf(left(uexp), Zeta: GammaAGamma, l(B), [SubPrf])) :-
      append(Gamma1, [l(!A, Phi, [])|Gamma2], GammaAGamma),
      append(Gamma1, Gamma2, GammaGamma).

% 20

pone(Zeta: Gamma, l(C lio B, 0), [Zeta: GammaB => _: SubPrf], prf(right(liover), Zeta: Gamma, l(C lio B), [SubPrf])) :-
      vector(B, _, Bvec),
      append(Gamma, [Bvec], GammaB).

% 21

pone(Zeta: Gamma, l(A liu C, Chi), [Zeta: [Avec|Gamma] => l(C, Chi): SubPrf], prf(right(liunder), Zeta: Gamma, l(A liu C), [SubPrf])) :-
      vector(A, _, Avec).

% 22

pone(Zeta: Gamma, l(C rio B, Chi), [Zeta: GammaB => l(C, Chi): SubPrf], prf(right(riover), Zeta: Gamma, l(C rio B), [SubPrf])) :-
      vector(B, _, Bvec),
      append(Gamma, [Bvec], GammaB).

% 23

pone(Zeta: Gamma, l(A riu C, 0), [Zeta: [Avec|Gamma] => l(C, _): SubPrf], prf(right(riunder), Zeta: Gamma, l(A riu C), [SubPrf])) :-
      vector(A, _, Avec).

% 24

pone(OMEGA, l(D, Omega), [OMEGAPr => l(D, Omega): SubPrf], prf(left(liproduct), OMEGA, l(D), [SubPrf])) :-
      subconfigofzoneandapp(OMEGA, OMEGAPr, H, Deltal, [l(A lip B, Chi, Ls)], Deltar),
      ssort(A, S),
      kappend(S, Ls1, Ls2, Ls),
      append(Deltal, [l(A, _, Ls1), l(B, Chi, Ls2)|Deltar], H).

% 25

pone(OMEGA, l(D, Omega), [OMEGAPr => l(D, Omega): SubPrf], prf(left(riproduct), OMEGA, l(D), [SubPrf])) :-
      subconfigofzoneandapp(OMEGA, OMEGAPr, H, Deltal, [l(A rip B, Chi, Ls)], Deltar),
      ssort(A, S),
      kappend(S, Ls1, Ls2, Ls),
      append(Deltal, [l(A, Chi, Ls1), l(B, _, Ls2)|Deltar], H).

% 26+

pone(Zeta: Gamma, l(C uic B, Chi), [Zeta: GammaPr => l(C, Chi): SubPrf], prf(right(uicircum), Zeta: Gamma, l(C uic B), [SubPrf])) :-
      vector(B, _, Bvec),
      configsort(Gamma, S),
      do_ones(S, [[1]|Ls]),
      fold(S, Gamma, [[Bvec]|Ls], GammaPr).

% 26-

pone(Zeta: Gamma, l(C uicn B, Chi), [Zeta: GammaPr => l(C, Chi): SubPrf], prf(right(uicircumn), Zeta: Gamma, l(C uicn B), [SubPrf])) :-
      vector(B, _, Bvec),
      configsort(Gamma, S),
      do_ones(S, [[1]|Ls]),
      append(Ls, [[Bvec]], LsB),
      fold(S, Gamma, LsB, GammaPr).

% 27+

pone(Zeta: Gamma, l(A uii C, 0), [Zeta: GammaPr => l(C, _): SubPrf], prf(right(uiinfix), Zeta: Gamma, l(A uii C), [SubPrf])) :-
```

```
    vector(A, _, Avec),
    ssort(A, S),
    do_ones(S, [[1]|Ls]),
    fold(S, Avec, [[Gamma]|Ls], GammaPr).

% 27-

pone(Zeta: Gamma, 1(A uiin C, 0), [Zeta: GammaPr => 1(C, _): SubPrf], prf(right(uiinfixn), Zeta: Gamma, 1(A uiin C), [SubPrf])) :-
    vector(A, _, Avec),
    ssort(A, S),
    do_ones(S, [[1]|Ls]),
    append(Ls, Gamma, LsGamma),
    fold(S, Avec, LsGamma, GammaPr).

% 28+

pone(Zeta: Gamma, 1(C lic B, Chi), [Zeta: GammaPr => 1(C, Chi): SubPrf], prf(right(licircum), Zeta: Gamma, 1(C lic B), [SubPrf])) :-
    vector(B, _, Bvec),
    configsort(Gamma, S),
    do_ones(S, [[1]|Ls]),
    fold(S, Gamma, [[Bvec]|Ls], GammaPr).

% 28-

pone(Zeta: Gamma, 1(C licn B, Chi), [Zeta: GammaPr => 1(C, Chi): SubPrf], prf(right(licircumn), Zeta: Gamma, 1(C licn B), [SubPrf])) :-
    vector(B, _, Bvec),
    configsort(Gamma, S),
    do_ones(S, [[1]|Ls]),
    append(Ls, [[Bvec]], LsB),
    fold(S, Gamma, LsB, GammaPr).

% 29+

pone(Zeta: Gamma, 1(A lii C, 0), [Zeta: GammaPr => 1(C, 0): SubPrf], prf(right(liinfix), Zeta: Gamma, 1(A lii C), [SubPrf])) :-
    vector(A, _, Avec),
    ssort(A, S),
    do_ones(S, [[1]|Ls]),
    fold(S, Avec, [[Gamma]|Ls], GammaPr).

% 29-

pone(Zeta: Gamma, 1(A liin C, 0), [Zeta: GammaPr => 1(C, 0): SubPrf], prf(right(liinfixn), Zeta: Gamma, 1(A liin C), [SubPrf])) :-
    vector(A, _, Avec),
    ssort(A, S),
    do_ones(S, [[1]|Ls]),
    append(Ls, Gamma, LsGamma),
    fold(S, Avec, LsGamma, GammaPr).

% 30+

pone(OMEGA, 1(D, Omega), [OMEGAPr => 1(D, Omega): SubPrf], prf(left(uidprod), OMEGA, 1(D), [SubPrf])) :-
    subconfigofzoneandapp(OMEGA, OMEGAPr, H, Delta1, [1(A uidp B, Chi, Ls)], Deltar),
    ssort(B, S),
    kappend(S, Ls1, Ls2, Ls),
    append(Delta1, [1(A, _, [[1(B, Chi, Ls1)]|Ls2])|Deltar], H).

% 30-

pone(OMEGA, 1(D, Omega), [OMEGAPr => 1(D, Omega): SubPrf], prf(left(uidprodn), OMEGA, 1(D), [SubPrf])) :-
    subconfigofzoneandapp(OMEGA, OMEGAPr, H, Delta1, [1(A uidpn B, Chi, Ls)], Deltar),
    ssort(B, S),
    kappend(S, Ls1, Ls2, Ls),
    append(Ls2, [[1(B, Chi, Ls1)]], Ls2B),
    append(Delta1, [1(A, _, Ls2B)|Deltar], H).

% 31+

pone(OMEGA, 1(D, Omega), [OMEGAPr => 1(D, Omega): SubPrf], prf(left(lidprod), OMEGA, 1(D), [SubPrf])) :-
    subconfigofzoneandapp(OMEGA, OMEGAPr, H, Delta1, [1(A lidp B, Chi, Ls)], Deltar),
    ssort(B, S),
```

```
      kappend(S, Ls1, Ls2, Ls),
      append(Deltal, [l(A, Chi, [[l(B, _, Ls1)]|Ls2])|Deltar], H).

% 31-

pone(OMEGA, l(D, Omega), [OMEGAPr => l(D, Omega): SubPrf], prf(left(lidprodn), OMEGA, l(D), [SubPrf])) :-
      subconfigofzoneandapp(OMEGA, OMEGAPr, H, Deltal, [l(A dpn B, Chi, Ls)], Deltar),
      ssort(B, S),
      kappend(S, Ls1, Ls2, Ls),
      append(Ls2, [[l(B, _, Ls1)]], Ls2B),
      append(Deltal, [l(A, Chi, Ls2B)|Deltar], H).

% 32

pone(OMEGA, l(A, Phi), [OMEGAPr => l(A, Phi): SubPrf], prf(left(words), OMEGA, l(A), [SubPrf])) :-
      subconfigofzoneandapp(OMEGA, OMEGAPr, H, Deltal, [l(w([]), _, [])], Deltar),
      append(Deltal, Deltar, H).

% 33

pone(OMEGA, l(A iac B, Chi), [OMEGA => l(A, Chi): SubPrf1, OMEGA => l(B, Chi): SubPrf2], prf(right(iaconj), OMEGA, l(A iac B), [SubPrf1, SubPrf2])).

% 34

pone(OMEGA, l(C, Chi), [OMEGAPr1 => l(C, Chi): SubPrf1, OMEGAPr2 => l(C, Chi): SubPrf2], prf(left(iadisj), OMEGA, l(C), [SubPrf1, SubPrf2])) :-
      leafofsubconfigofzone(OMEGA, OMEGAPr, xxx, l(A iad B, Z, Ls)),
      leafofsubconfigofzone(OMEGAPr, OMEGAPr1, l(A, Z, Ls), xxx),
      leafofsubconfigofzone(OMEGAPr, OMEGAPr2, l(B, Z, Ls), xxx).

% 35

pone(OMEGA, l(V iu A, Phi), [OMEGA => l(A1, Phi): SubPrf], prf(right(iunivq), OMEGA, l(V iu A), [SubPrf])) :-
      gensymb(S),
      tsubst(S, V, A, A1).

% 36

pone(OMEGA, l(B, Psi), [OMEGAPr => l(B, Psi): SubPrf], prf(left(iexstq), OMEGA, l(B), [SubPrf])) :-
      leafofsubconfigofzone(OMEGA, OMEGAPr, l(A1, X, Ls), l(V ie A, X, Ls)),
      gensymb(S),
      tsubst(S, V, A, A1).

% 37

pone(OMEGA, l('iL'A, Phi), [OMEGA => l(A, Phi): SubPrf], prf(right(iumod), OMEGA, l('iL'A), [SubPrf])) :-
      allumodszone(OMEGA).

% 38

pone(OMEGA, l('M'B, Psi), [OMEGAPr => l('M'B, Psi): SubPrf], prf(left(uemod), OMEGA, l('M'B), [SubPrf])) :-
      leafofsubconfigofzone(OMEGA, OMEGAPr, H, l('M'A, Chi, Ls)),
      allumodszone(OMEGAPr),
      H = l(A, Chi, Ls).

pone(OMEGA, l(iM B, Psi), [OMEGAPr => l('M'B, Psi): SubPrf], prf(left(uemod), OMEGA, l('iM'B), [SubPrf])) :-
      leafofsubconfigofzone(OMEGA, OMEGAPr, H, l('M'A, Chi, Ls)),
      allumodszone(OMEGAPr),
      H = l(A, Chi, Ls).

% 39

pone(Zeta: Gamma, l(lp A, Phi), [Zeta: Gamma1 => l(A, Phi): SubPrf], prf(right(lproj), Zeta: Gamma, l(lp A), [SubPrf])) :-
      append(Gamma, [1], Gamma1).

% 40

pone(Zeta: Gamma, l(rp A, Phi), [Zeta: [1|Gamma] => l(A, Phi): SubPrf], prf(right(rproj), Zeta: Gamma, l(rp A), [SubPrf])).

% 41
```

```
pone(OMEGA, l(B, Psi), [OMEGAPr => l(B, Psi): SubPrf], prf(left(linj), OMEGA, l(B), [SubPrf])) :-
    subconfigofzoneandapp(OMEGA, OMEGAPr, H, Delta1, [l(li A, Phi, Ls)], Deltar),
    append(Delta1, [l(A, Phi, Ls), l|Deltar], H).

% 42

pone(OMEGA, l(B, Psi), [OMEGAPr => l(B, Psi): SubPrf], prf(left(rinj), OMEGA, l(B), [SubPrf])) :-
    subconfigofzoneandapp(OMEGA, OMEGAPr, H, Delta1, [l(ri A, Phi, Ls)], Deltar),
    append(Delta1, [l, l(A, Phi, Ls)|Deltar], H).

% 43+

pone(Zeta: Delta, l(sp B, Psi), [Zeta: DeltaPr => l(B, Psi): SubPrf], prf(right(split), Zeta: Delta, l(sp B), [SubPrf])) :-
    configsort(Delta, [1|S]),
    do_ones(S, Ls),
    fold([1|S], Delta, [[]|Ls], DeltaPr).

% 43-

pone(Zeta: Delta, l(spn B, Psi), [Zeta: DeltaPr => l(B, Psi): SubPrf], prf(right(splitn), Zeta: Delta, l(spn B), [SubPrf])) :-
    configsort(Delta, [1|S]),
    do_ones(S, Ls),
    append(Ls, [[], Lss),
    fold([1|S], Delta, Lss, DeltaPr).

% 44+

pone(OMEGA, l(C, Chi), [OMEGAPr => l(C, Chi): SubPrf], prf(left(bridge), OMEGA, l(C), [SubPrf])) :-
    leafofsubconfigofzone(OMEGA, OMEGAPr, l(B, Psi, [[]|Ls]), l(bg B, Psi, Ls)).

% 44-

pone(OMEGA, l(C, Chi), [OMEGAPr => l(C, Chi): SubPrf], prf(left(bridgen), OMEGA, l(C), [SubPrf])) :-
    leafofsubconfigofzone(OMEGA, OMEGAPr, l(B, Psi, Ls), l(bgn B, Psi, Ls)),
    append(Ls, [[]], Lss).

% 45

pone(Zeta: Gamma, l(C nd A, [lnd, X, Chi]), [Zeta: [Avec|Gamma] => l(C, Chi): SubPrf1, Zeta: GammaA => l(C, Chi): SubPrf2], prf(right(ndiv), Zeta: Gamma, l(C nd A), [SubPrf1, SubPrf2])) :-
    vector(A, X, Avec),
    append(Gamma, [Avec], GammaA).

% 46

pone(OMEGA, l(D, Omega), [OMEGAPr => l(D, Omega): SubPrf1, OMEGAPrPr => l(D, Omega): SubPrf2], prf(left(nprod), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
    subconfigofzoneandapp(OMEGA, OMEGAPr, H, Delta1, [l(A np B, Chi, LsALsB)], Deltar),
    makecopy((OMEGAPr, H, Delta1, Deltar), (OMEGAPrPr, HPr, Delta1Pr, DeltarPr)),
    ssort(A, SA),
    kappend(SA, LsA, LsB, LsALsB),
    append(Delta1, [l(A, [fst, Chi], LsA), l(B, [snd, Chi], LsB)|Deltar], H),
    append(Delta1Pr, [l(B, [snd, Chi], LsB), l(A, [fst, Chi], LsA)|DeltarPr], HPr).

% 47

pone(Zeta: Gamma, l(C nci B, [lnd, Y, Chi]), [Zeta: GammaPr => l(C, Chi): SubPrf1, Zeta: GammaPrPr => l(C, Chi): SubPrf2], prf(right(ninfix), Zeta: Gamma, l(C nci B), [SubPrf1, SubPrf2])) :-
    vector(B, Y, Bvec),
    configsort(Gamma, [1|S]),
    do_ones(S, Ls),
    fold(Gamma, [1|S], [[Bvec]|Ls], GammaPr),
    append(Ls, [[Bvec]], LsB),
    fold(Gamma, [1|S], LsB, GammaPrPr).

% 48

pone(Zeta: Gamma, l(A nin C, [lnd, X, Chi]), [Zeta: GammaPr => l(C, Chi): SubPrf1, Zeta: GammaPrPr => l(C, Chi): SubPrf2], prf(right(ninfix), Zeta: Gamma, l(A nin C), [SubPrf1, SubPrf2])) :-
    vector(A, X, Avec),
    ssort(A, [1|S]),
    do_ones(S, Ls),
    fold([Avec], [1|S], [Gamma|Ls], GammaPr),
    append(Ls, Gamma, LsGamma),
```

```
      fold([Avec], [1|S], LsGamma, GammaPrPr).

% 49

pone(OMEGA, l(D, Omega), [OMEGAPr => l(D, Omega): SubPrf1, OMEGAPrPr => l(D, Omega): SubPrf2], prf(left(ndprod), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      subconfigofzoneandapp(OMEGA, OMEGAPr, H, Deltal, [l(A ndp B, Chi, Ls)], Deltar),
      makecopy((OMEGAPr, H, Deltal, Deltar), (OMEGAPrPr, HPr, DeltalPr, DeltarPr)),
      ssort(B, SB),
      kappend(SB, Ls1B, Ls1A, Ls),
      append(Deltal, [l(A, [fst, Chi], [l(B, [snd, Chi], Ls1B)|Ls1A])|Deltar], H),
      ssort(A, [1|S]),
      kappend(S, Ls2A, Ls2B, Ls),
      append(Ls2A, [[l(B, [snd, Chi], Ls2B)]], Lss),
      append(DeltalPr, [l(A, [fst, Chi], Lss)|DeltarPr], HPr).

makecopy(X, Y) :- %X=Y.
      assert(copy(X)),
      retract(copy(Y)).

% ssort(+A, -SA) means that type A is of sort SA where sort n
% is represented as a list of n 1's.

ssort(P, SP) :-
      primitive(P, SP).

ssort(C/B, S) :-          % 1
      ssort(B, S1),
      ssort(C, S2),
      append(S, S1, S2).

ssort(A bs C, S) :-       % 2
      ssort(A, S1),
      ssort(C, S2),
      append(S1, S, S2).

ssort(A*B, S) :-          % 3
      ssort(A, S1),
      ssort(B, S2),
      append(S1, S2, S).

ssort(i, []).             % 4

ssort(C ci B, [1|S]) :-   % 5+
      ssort(B, S1),
      ssort(C, S2),
      append(S, S1, S2).

ssort(C cin B, [1|S]) :-  % 5-
      ssort(B, S1),
      ssort(C, S2),
      append(S, S1, S2).

ssort(A in C, S) :-       % 6+
      ssort(A, [1|S1]),
      ssort(C, S2),
      append(S1, S, S2).

ssort(A inn C, S) :-      % 6-
      ssort(A, [1|S1]),
      ssort(C, S2),
      append(S1, S, S2).

ssort(A dp B, S) :-       % 7+
      ssort(A, [1|S1]),
      ssort(B, S2),
      append(S1, S2, S).

ssort(A dpn B, S) :-      % 7-
      ssort(A, [1|S1]),
      ssort(B, S2),
```

```
    append(S1, S2, S).

ssort(j, [1]).                          % 8

ssort(A&B, S) :-                        % 9
        ssort(A, S),
        ssort(B, S).

ssort(A+B, S) :-                        % 10
        ssort(A, S),
        ssort(B, S).

ssort(_ u A, S) :-                      % 11
        ssort(A, S).

ssort(_ e A, S) :-                      % 12
        ssort(A, S).

ssort('L'A, S) :-                       % 13
        ssort(A, S).

ssort('M'A, S) :-                       % 14
        ssort(A, S).

ssort(ab A, S) :-                       % 15
        ssort(A, S).

ssort(br A, S) :-                       % 16
        ssort(A, S).

ssort('|'A, []) :-                      % 17
        ssort(A, []).

ssort('?'A, []) :-                      % 18
        ssort(A, []).

ssort(B lca A, S) :-                    % 19
        ssort(B, S),
        ssort(A, SPr),
        append(SPr, _, S).

ssort(C lio B, S) :-                    % 20
        ssort(B, S1),
        ssort(C, S2),
        append(S, S1, S2).

ssort(A liu C, S) :-                    % 21
        ssort(A, S1),
        ssort(C, S2),
        append(S1, S, S2).

ssort(C rio B, S) :-                    % 22
        ssort(B, S1),
        ssort(C, S2),
        append(S, S1, S2).

ssort(A riu C, S) :-                    % 23
        ssort(A, S1),
        ssort(C, S2),
        append(S1, S, S2).

ssort(A lip B, S) :-                    % 24
        ssort(A, S1),
        ssort(B, S2),
        append(S1, S2, S).

ssort(A rip B, S) :-                    % 25
        ssort(A, S1),
        ssort(B, S2),
        append(S1, S2, S).
```

```
ssort(C uic B, [1|S]) :-  % 26+
      ssort(B, S1),
      ssort(C, S2),
      append(S, S1, S2).

ssort(C uicn B, [1|S]) :- % 26-
      ssort(B, S1),
      ssort(C, S2),
      append(S, S1, S2).

ssort(A uii C, S) :-      % 27+
      ssort(A, [1|S1]),
      ssort(C, S2),
      append(S1, S, S2).

ssort(A uiin C, S) :-     % 27-
      ssort(A, [1|S1]),
      ssort(C, S2),
      append(S1, S, S2).

ssort(C lic B, [1|S]) :-  % 28+
      ssort(B, S1),
      ssort(C, S2),
      append(S, S1, S2).

ssort(C licn B, [1|S]) :- % 28-
      ssort(B, S1),
      ssort(C, S2),
      append(S, S1, S2).

ssort(A lii C, S) :-      % 29+
      ssort(A, [1|S1]),
      ssort(C, S2),
      append(S1, S, S2).

ssort(A liin C, S) :-     % 29-
      ssort(A, [1|S1]),
      ssort(C, S2),
      append(S1, S, S2).

ssort(A uidp B, S) :-     % 30+
      ssort(A, [1|S1]),
      ssort(B, S2),
      append(S1, S2, S).

ssort(A uidpn B, S) :-    % 30-
      ssort(A, [1|S1]),
      ssort(B, S2),
      append(S1, S2, S).

ssort(A lidp B, S) :-     % 31+
      ssort(A, [1|S1]),
      ssort(B, S2),
      append(S1, S2, S).

ssort(A lidpn B, S) :-    % 32-
      ssort(A, [1|S1]),
      ssort(B, S2),
      append(S1, S2, S).

% ssort(w(_), []).          % 32

ssort(A iac B, S) :-      % 33
      ssort(A, S),
      ssort(B, S).

ssort(A iad B, S) :-      % 34
      ssort(A, S),
      ssort(B, S).
```

```
ssort(_ iu A, S) :-      % 35
    ssort(A, S).

ssort(_ ie A, S) :-      % 36
    ssort(A, S).

ssort(iL A, S) :-        % 37
    ssort(A, S).

ssort(iM A, S) :-        % 38
    ssort(A, S).

ssort(lp A, S) :-        % 39
    ssort(A, [1|S]).

ssort(rp A, S) :-        % 40
    ssort(A, [1|S]).

ssort(li A, [1|S]) :-    % 41
    ssort(A, S).

ssort(ri A, [1|S]) :-    % 42
    ssort(A, S).

ssort(sp A, [1|S]) :-    % 43+
    ssort(A, S).

ssort(spn A, [1|S]) :-   % 43-
    ssort(A, S).

ssort(bg A, S) :-        % 44+
    ssort(A, [1|S]).

ssort(bgn A, S) :-       % 44-
    ssort(A, [1|S]).

ssort(B nd A, S) :-      % 45
    ssort(A, S1),
    ssort(B, S2),
    append(S1, S, S2).

ssort(A np B, S) :-      % 46
    ssort(A, S1),
    ssort(B, S2),
    append(S1, S2, S).

ssort(C nci B, [1|S]) :- % 47
    ssort(B, S1),
    ssort(C, S2),
    append(S, S1, S2).

ssort(A nin C, S) :-     % 48
    ssort(A, [1|S1]),
    ssort(C, S2),
    append(S1, S, S2).

ssort(A ndp B, S) :-     % 49
    ssort(A, [1|S1]),
    ssort(B, S2),
    append(S1, S2, S).

ssort(A-B, S) :-         % 50
    ssort(A, S),
    ssort(B, S).

% subconfigandapp(+Delta, -DeltaPr, -H, -Deltal, -Ls, -Deltar) means that configuration Delta
% can be analysed into DeltaPr(Deltal, Ls, Deltar) where variable H is a leaf in
% DeltaPr marking the position of maximal subconfiguration Deltal, Ls, Deltar in Delta
```

```
subconfigandapp(Delta, DeltaPr, H, Deltal, Ls, Deltar) :-
      subconfig(Delta, DeltaPr, H, Gamma),
      nappend([Deltal, Ls, Deltar], Gamma).

% subconfigofzoneandapp(+Zeta, -ZetaPr, -H, -Deltal, -Ls, -Deltar) means that zone Zeta
% can be analysed into ZetaPr(Deltal, Ls, Deltar) where variable H is a leaf in
% ZetaPr marking the position of maximal subconfiguration Deltal, Ls, Deltar in Zeta

subconfigofzoneandapp(Zeta, ZetaPr, H, Deltal, Ls, Deltar) :-
      subconfigofzone(Zeta, ZetaPr, H, Delta),
      nappend([Deltal, Ls, Deltar], Delta).

% subconfig(+Delta, -DeltaPr, -H, -Gamma) means that
% configuration Delta can be analysed into DeltaPr(Gamma)
% where H is a leaf variable in DeltaPr marking the
% position of subconfiguration Gamma in Zeta.

subconfig(Delta, H, H, Delta).

subconfig(Delta, Delta1, H, Delta2) :-
      append(Deltaleft, [l(X, Phi, Lss)|Deltaright], Delta),
      append(L1, [Ls|L2], Lss),
      subconfig(Ls, LsPr, H, Delta2),
      append(L1, [LsPr|L2], LssPr),
      append(Deltaleft, [l(X, Phi, LssPr)|Deltaright], Delta1).

subconfig(Delta, Delta1, H, Delta2) :-
      append(Deltaleft, [b(Zeta)|Deltaright], Delta),
      subconfigofzone(Zeta, ZetaPr, H, Delta2),
      append(Deltaleft, [b(ZetaPr)|Deltaright], Delta1).

% leafofsubconfig(+Delta, -DeltaPr, -H, ?L) means that configuration Delta can be
% analysed into DeltaPr(L) where H is a leaf variable in DeltaPr marking the
% position of leaf L in Delta

leafofsubconfig(Delta, DeltaPr, H, L) :-
      subconfigandapp(Delta, DeltaPr, H1, Deltal, [L], Deltar),
      append(Deltal, [H|Deltar], H1).

% leafofsubconfigofzone(+Zeta, -ZetaPr, -H, ?L) means that zone Zeta can be
% analysed into ZetaPr(L) where H is a leaf variable in ZetaPr marking the
% position of leaf L in Delta

leafofsubconfigofzone(Omega: Delta, Omega: DeltaPr, H, L) :-
      subconfigandapp(Delta, DeltaPr, H1, Deltal, [L], Deltar),
      append(Deltal, [H|Deltar], H1).

% subconfigofzoneP(+Zeta, -ZetaPr, H, Delta) means that Zeta can be analysed into
% ZetaPr(Delta) where variable H is a leaf in ZetaPr marking the position of
% maximal subconfiguration Delta in Zeta

subconfigofzone(Omega: Delta, Omega: DeltaPr, H, Gamma) :-
      subconfig(Delta, DeltaPr, H, Gamma).

% subzoneandapp(+Zeta, -ZetaPr, -H, -Stoup, L1, L2, L3) means that zone Zeta can be
% analyzed into ZetaPr(Stoup: L1, L2, L3) where variable H in ZetaPr marks the
% position of subzone Stoup: L1, L2, L3

subzoneandapp(Zeta, ZetaPr, H, Stoup, L1, L2, L3) :-
      subzone(Zeta, ZetaPr, H, Stoup: Lss),
      nappend([L1, L2, L3], Lss).

% subzone(+Zeta, -ZetaPr, -H, -ZetaPrPr) means that zone Zeta can be analysed
% into ZetaPr(ZetaPrPr) where H is a variable in ZetaPr marking the position of
% subzone ZetaPrPr in Zeta

subzone(Zeta, H, H, Zeta).

subzone(Omega: Delta, Omega: DeltaPr, H, Zeta) :-
      subzoneofconfig(Delta, DeltaPr, H, Zeta).
```

```
% subzoneofconfig(+Delta, -DeltaPr, -H, -Zeta) means that configuration Delta
% can be analysed into DeltaPr(Zeta) where variable H is a leaf in DeltaPr
% marking the position of subzone Zeta in Delta.

subzoneofconfig(Delta, DeltaPr, H, Zeta) :-
      append(Deltal, [l(A, Phi, Lss)|Deltar], Delta),
         append(Lsl, [Ls|Ls2], Lss),
         subzoneofconfig(Ls, LsPr, H, Zeta),
         append(Lsl, [LsPr|Ls2], LssPr),
         append(Deltal, [l(A, Phi, LssPr)|Deltar], DeltaPr).

subzoneofconfig(Delta, DeltaPr, H, Zeta) :-
      append(Deltal, [b(ZetaPr)|Deltar], Delta),
         subzone(ZetaPr, ZetaPrPr, H, Zeta),
         append(Deltal, [b(ZetaPrPr)|Deltar], DeltaPr).

% vector(+A, +Phi, -Avec) means that Avec is the vector of type A with semantics Phi

vector(A, Phi, l(A, Phi, Ls)) :-
      ssort(A, S),
      do_ones(S, Ls).

% do_ones(+S, -Ls) means that where S is a sort in unary notation,
% Ls is the list of singleton lists of separators of the same length.

do_ones([],[]).

do_ones([1|L], [[1]|L1]) :-
      do_ones(L, L1).

% nappend(?Ls, ?L) means that L is the concatenation of the
% list of lists Ls.

nappend([], []).

nappend([[]|Ls], L) :-
      nappend(Ls, L).

nappend([[H|L1]|Ls], [H|L]) :-
      nappend([L1|Ls], L).

% partition(+L, -L1, -L2) means that L1 and L2 are a partition of L

partition([], [], []).

partition([H|T], [H|T1], T2) :-
      partition(T, T1, T2).

partition([H|T], T1, [H|T2]) :-
   partition(T, T1, T2).

% p1(+OMEGA, +APhi, -Prf) means that OMEGA => APhi is provable with proof Prf

p1(OMEGA, l(A, Phi), Prf) :-
      countcheck(OMEGA, A),
      p1lst([OMEGA => l(A, Phi): Prf], L),
      p2lst(L).

% p1lst(+L1, -L2) means that the list of sequents L1 unfolds don't care
% nondeterministically to the list of sequents L2 by asynchronous/invertible/"phase 1"
% rules

p1lst([], []).

p1lst([OMEGA => APhiPrf|L], LPrPr) :-
      leafofsubconfigofzone(OMEGA, _, _, l(W), _, _),
         var(W), !, W = [_|_]),
         p1lst([OMEGA => APhiPrf|L], LPrPr).
```

```
p1lst([OMEGA => APhi: Prf|L], LPrPr) :-
      pone(OMEGA, APhi, Ls, Prf), (!),
      append(Ls, L, LPr),
      p1lst(LPr, LPrPr).

p1lst([H|T], [H|L]) :-
      p1lst(T, L).

% p2lst(+L) means that the sequents of list L are provable starting with
% synchronous/noninvertible/"phase 2" rules

p2lst([]).

p2lst([OMEGA => APhi: Prf|L]) :-
      p2(OMEGA, APhi, Prf),
      p2lst(L).

% p2(+OMEGA, +DOmega, -Prf) means that sequent OMEGA => DOmega is derivable with
% proof Prf

p2(OMEGA, l(D, Omega), Prf) :-
      countcheck(OMEGA, D),
      p2aux(OMEGA, l(D, Omega), Prf).

p2aux(OMEGA, l(A, Phi), Prf) :-
      p2r(OMEGA, f(A, Phi), Prf).

p2aux(OMEGA, DOmega, Prf) :-
      leafofsubconfigofzone(OMEGA, OMEGAPr, f(A, Phi, Ls), l(A, Phi, Ls)),
      p2l(A, OMEGAPr, DOmega, Prf).

p2aux(OMEGA, DOmega, Prf) :-
      subzone(OMEGA, OMEGAPr, StoupPr: Delta, [H|T]: Delta),
      append(Stoup1, [l(A, Phi, [])|Stoup2], [H|T]),
      append(Stoup1, [f(A, Phi, [])|Stoup2], StoupPr),
      p2s(OMEGAPr, DOmega, Prf).

% p2l(+A, +OMEGA, -DOmega -Prf) means that sequent OMEGA => DOmega, in which type A is
% focused in its configuration, is provable with proof Prf

p2l(A, []: [f(A, Phi, [])], l(A, Phi), prf(id, []: [f(A, Phi, [])], l(A), [])) :-
      atfoc(inp),
      primitive(A, []), (!).

p2l(A, []: [f(A, Phi, [[1]])], l(A, Phi), prf(id, []: [f(A, Phi, [[1]])], l(A), [])) :-
      atfoc(inp),
      primitive(A, [1]), (!).

% 1

p2l(P/Q, OMEGA, l(D, Omega), prf(left(over), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(P, inp), pos(Q, out),
      ssort(Q, SQ), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(P/Q, Psi, Ls2)|Ls], Deltar),
      partition(Stoup, Stoup1, Stoup2),
      fold(SQ, Gamma, Ls1, Ls),
      p2r(Stoup1: Gamma, f(Q, Phi), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [f(P, [app, Psi, Phi], Lss)|Deltar], Delta),
      p2l(P, OMEGAPr, l(D, Omega), SubPrf2).

p2l(P/N, OMEGA, l(D, Omega), prf(left(over), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(P, inp), neg(N, out),
      ssort(N, SN), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(P/N, Psi, Ls2)|Ls], Deltar),
      partition(Stoup, Stoup1, Stoup2),
      fold(SN, Gamma, Ls1, Ls),
      p1(Stoup1: Gamma, l(N, Phi), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [f(P, [app, Psi, Phi], Lss)|Deltar], Delta),
```

```
    p2l(P, OMEGAPr, l(D, Omega), SubPrf2).

p2l(N/P, OMEGA, l(D, Omega), prf(left(over), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
        neg(N, inp), pos(P, out),
        ssort(P, SP), (!),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(N/P, Psi, Ls2)|Ls], Deltar),
        partition(Stoup, Stoup1, Stoup2),
        fold(SP, Gamma, Ls1, Ls),
        p2r(Stoup1: Gamma, f(P, Phi), SubPrf1),
        append(Ls1, Ls2, Lss),
        append(Deltal, [l(N, [app, Psi, Phi], Lss)|Deltar], Delta),
        p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(N/M, OMEGA, l(D, Omega), prf(left(over), OMEGA, l(D), [SubPrf1, SubPrf2])) :- (!),
        neg(N, inp), neg(M, out),
        ssort(M, SM),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(N/M, Psi, Ls2)|Ls], Deltar),
        partition(Stoup, Stoup1, Stoup2),
        fold(SM, Gamma, Ls1, Ls),
        p1(Stoup1: Gamma, l(M, Phi), SubPrf1),
        append(Ls1, Ls2, Lss),
        append(Deltal, [l(N, [app, Psi, Phi], Lss)|Deltar], Delta),
        p1(OMEGAPr, l(D, Omega), SubPrf2).

    % 2

p2l(P bs Q, OMEGA, l(D, Omega), prf(left(under), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
        pos(P, out), pos(Q, inp), (!),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, LsB, Deltar),
        append(Ls, [f(P bs Q, Psi, Ls2)], LsB),
        ssort(P, SP),
        partition(Stoup, Stoup1, Stoup2),
        fold(SP, Gamma, Ls1, Ls),
        p2r(Stoup1: Gamma, f(P, Phi), SubPrf1),
        append(Ls1, Ls2, Lss),
        append(Deltal, [f(Q, [app, Psi, Phi], Lss)|Deltar], Delta),
        p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(P bs N, OMEGA, l(D, Omega), prf(left(under), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
        pos(P, out), neg(N, inp), (!),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, LsB, Deltar),
        append(Ls, [f(P bs N, Psi, Ls2)], LsB),
        ssort(P, SP),
        partition(Stoup, Stoup1, Stoup2),
        fold(SP, Gamma, Ls1, Ls),
        p2r(Stoup1: Gamma, f(P, Phi), SubPrf1),
        append(Ls1, Ls2, Lss),
        append(Deltal, [l(N, [app, Psi, Phi], Lss)|Deltar], Delta),
        p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(N bs P, OMEGA, l(D, Omega), prf(left(under), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
        neg(N, out), pos(P, inp), (!),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, LsB, Deltar),
        append(Ls, [f(N bs P, Psi, Ls2)], LsB),
        ssort(N, SN),
        partition(Stoup, Stoup1, Stoup2),
        fold(SN, Gamma, Ls1, Ls),
        p1(Stoup1: Gamma, l(N, Phi), SubPrf1),
        append(Ls1, Ls2, Lss),
        append(Deltal, [f(P, [app, Psi, Phi], Lss)|Deltar], Delta),
        p2l(P, OMEGAPr, l(D, Omega), SubPrf2).

p2l(N bs M, OMEGA, l(D, Omega), prf(left(under), OMEGA, l(D), [SubPrf1, SubPrf2])) :- (!),
        neg(N, out), neg(N, inp),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, LsB, Deltar),
        append(Ls, [f(N bs M, Psi, Ls2)], LsB),
        ssort(N, SN),
        partition(Stoup, Stoup1, Stoup2),
        fold(SN, Gamma, Ls1, Ls),
        p1(Stoup1: Gamma, l(N, Phi), SubPrf1),
```

```
      append(Ls1, Ls2, Lss),
      append(Deltal, [l(M, [app, Psi, Phi], Lss)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

% 5+

p2l(Q ci P, OMEGA, l(D, Omega), prf(left(circumfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), pos(P, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q ci P, Phi, [Gamma|Gammas])], Deltar), (!),
      partition(Stoup, Stoup1, Stoup2),
      p2r(Stoup1: Gamma, f(P, Psi), SubPrf1),
      ssort(P, SP),
      do_ones(SP, Ls),
      append(Ls, Gammas, LsGammas),
      append(Deltal, [f(Q, [app, Phi, Psi], LsGammas)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(Q ci N, OMEGA, l(D, Omega), prf(left(circumfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), neg(N, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q ci N, Phi, [Gamma|Gammas])], Deltar), (!),
      partition(Stoup, Stoup1, Stoup2),
      p1(Stoup1: Gamma, l(N, Psi), SubPrf1),
      ssort(N, SN),
      do_ones(SN, Ls),
      append(Ls, Gammas, LsGammas),
      append(Deltal, [f(Q, [app, Phi, Psi], LsGammas)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(M ci P, OMEGA, l(D, Omega), prf(left(circumfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(M, inp), pos(P, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M ci P, Phi, [Gamma|Gammas])], Deltar), (!),
      partition(Stoup, Stoup1, Stoup2),
      p2r(Stoup1: Gamma, f(P, Psi), SubPrf1),
      ssort(P, SP),
      do_ones(SP, Ls),
      append(Ls, Gammas, LsGammas),
      append(Deltal, [l(M, [app, Phi, Psi], LsGammas)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(M ci N, OMEGA, l(D, Omega), prf(left(circumfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(M, inp), neg(N, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M ci N, Phi, [Gamma|Gammas])], Deltar), (!),
      partition(Stoup, Stoup1, Stoup2),
      p1(Stoup1: Gamma, l(N, Psi), SubPrf1),
      ssort(N, SN),
      do_ones(SN, Ls),
      append(Ls, Gammas, LsGammas),
      append(Deltal, [l(M, [app, Phi, Psi], LsGammas)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

% 5-

p2l(Q cin P, OMEGA, l(D, Omega), prf(left(circumfixn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), pos(P, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q cin P, Phi, GammasGamma)], Deltar),
      append(Gammas, [Gamma], GammasGamma), (!),
      partition(Stoup, Stoup1, Stoup2),
      p2r(Stoup1: Gamma, f(P, Psi), SubPrf1),
      ssort(P, SP),
      do_ones(SP, Ls),
      append(Gammas, Ls, GammasLs),
      append(Deltal, [f(Q, [app, Phi, Psi], GammasLs)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(Q cin N, OMEGA, l(D, Omega), prf(left(circumfixn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), neg(N, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q cin N, Phi, GammasGamma)], Deltar),
      append(Gammas, [Gamma], GammasGamma), (!),
      partition(Stoup, Stoup1, Stoup2),
      p1(Stoup1: Gamma, l(N, Psi), SubPrf1),
```

```
    ssort(N, SN),
    do_ones(SN, Ls),
    append(Gammas, Ls, GammasLs),
    append(Deltal, [f(Q, [app, Phi, Psi], GammasLs)|Deltar], Delta),
    p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(M cin P, OMEGA, l(D, Omega), prf(left(circumfixn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
    neg(M, inp), pos(P, out),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M cin P, Phi, GammasGamma)], Deltar),
    append(Gammas, [Gamma], GammasGamma), (!),
    partition(Stoup, Stoup1, Stoup2),
    p2r(Stoup1: Gamma, f(P, Psi), SubPrf1),
    ssort(P, SP),
    do_ones(SP, Ls),
    append(Gammas, Ls, GammasLs),
    append(Deltal, [l(M, [app, Phi, Psi], GammasLs)|Deltar], Delta),
    p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(M cin N, OMEGA, l(D, Omega), prf(left(circumfixn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
    neg(M, inp), neg(N, out),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M cin N, Phi, GammasGamma)], Deltar),
    append(Gammas, [Gamma], GammasGamma), (!),
    partition(Stoup, Stoup1, Stoup2),
    p1(Stoup1: Gamma, l(M, Psi), SubPrf1),
    ssort(M, SM),
    do_ones(SM, Ls),
    append(Gammas, Ls, GammasLs),
    append(Deltal, [l(M, [app, Phi, Psi], GammasLs)|Deltar], Delta),
    p1(OMEGAPr, l(D, Omega), SubPrf2).

% 6+

p2l(P in Q, OMEGA, l(D, Omega), prf(left(infix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
    pos(P, out), pos(Q, inp), (!),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
    genins(Gamma1, [f(P in Q, Psi, Ls1)|_], [], L),
    ssort(P, SP),
    partition(Stoup, Stoup1, Stoup2),
    fold(SP, Gamma2, [[1]|Ls2], Gamma1),
    p2r(Stoup1: Gamma2, f(P, Phi), SubPrf1),
    append(Ls1, Ls2, Lss),
    append(Deltal, [f(Q, [app, Psi, Phi], Lss)|Deltar], Delta),
    p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(P in M, OMEGA, l(D, Omega), prf(left(infix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
    pos(P, out), neg(M, inp), (!),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
    genins(Gamma1, [f(P in M, Psi, Ls1)|_], [], L),
    ssort(P, SP),
    partition(Stoup, Stoup1, Stoup2),
    fold(SP, Gamma2, [[1]|Ls2], Gamma1),
    p2r(Stoup1: Gamma2, f(P, Phi), SubPrf1),
    append(Ls1, Ls2, Lss),
    append(Deltal, [f(P, Psi, Phi], Lss)|Deltar], Delta),
    p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(N in Q, OMEGA, l(D, Omega), prf(left(infix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
    neg(N, out), pos(Q, inp), (!),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
    genins(Gamma1, [f(N in Q, Psi, Ls1)|_], [], L),
    ssort(N, SN),
    partition(Stoup, Stoup1, Stoup2),
    fold(SN, Gamma2, [[1]|Ls2], Gamma1),
    p1(Stoup1: Gamma2, l(N, Phi), SubPrf1),
    append(Ls1, Ls2, Lss),
    append(Deltal, [f(Q, [app, Psi, Phi], Lss)|Deltar], Delta),
    p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(N in M, OMEGA, l(D, Omega), prf(left(infix), OMEGA, l(D), [SubPrf1, SubPrf2])) :- (!),
    neg(N, out), neg(M, inp),
```

```
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, [f(N in M, Psi, Ls1)|_], [], L),
      ssort(N, SN),
      partition(Stoup, Stoup1, Stoup2),
      fold(SN, Gamma2, [[1]|Ls2], Gamma1),
      p1(Stoup1: Gamma2, l(N, Phi), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [l(M, [app, Psi, Phi], Lss)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

% 6-

p2l(P inn Q, OMEGA, l(D, Omega), prf(left(infixn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(P, out), pos(Q, inp), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, Ls, [], L),
      append(_, [f(P inn Q, Psi, Ls1)], Ls),
      ssort(P, [1|N]),
      kappend(N, Ls2, [[1]], Ls3),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|N], Gamma2, Ls3, Gamma1),
      p2r(Stoup1: Gamma2, f(P, Phi), SubPrf1),
      append(Ls2, Ls1, Lss),
      append(Deltal, [f(Q, [app, Psi, Phi], Lss)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(P inn M, OMEGA, l(D, Omega), prf(left(infixn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(P, out), neg(M, inp), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, Ls, [], L),
      append(_, [f(P inn M, Psi, Ls1)], Ls),
      ssort(P, [1|N]),
      partition(Stoup, Stoup1, Stoup2),
      kappend(N, Ls2, [[1]], Ls3),
      fold([1|N], Gamma2, Ls3, Gamma1),
      p2r(Stoup1: Gamma2, f(P, Phi), SubPrf1),
      append(Ls2, Ls1, Lss),
      append(Deltal, [l(M, [app, Psi, Phi], Lss)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(N inn Q, OMEGA, l(D, Omega), prf(left(infixn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(N, out), pos(Q, inp), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, Ls, [], L),
      append(_, [f(N inn Q, Psi, Ls1)], Ls),
      ssort(N, [1|NN]),
      kappend(NN, Ls2, [[1]], Ls3),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|NN], Gamma2, Ls3, Gamma1),
      p1(Stoup1: Gamma2, l(N, Phi), SubPrf1),
      append(Ls2, Ls1, Lss),
      append(Deltal, [f(Q, [app, Psi, Phi], Lss)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(N inn M, OMEGA, l(D, Omega), prf(left(infixn), OMEGA, l(D), [SubPrf1, SubPrf2])) :- (!),
      neg(N, out), neg(M, inp),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, Ls, [], L),
      append(_, [f(N inn M, Psi, Ls1)], Ls),
      ssort(N, [1|NN]),
      kappend(NN, Ls2, [[1]], Ls3),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|NN], Gamma2, Ls3, Gamma1),
      p1(Stoup1: Gamma2, l(N, Phi), SubPrf1),
      append(Ls2, Ls1, Lss),
      append(Deltal, [l(M, [app, Psi, Phi], Lss)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

% 9
```

```
p2l(Q&_, OMEGA, l(D, Omega), prf(left(aconj), OMEGA, l(D), [SubPrf])) :-
        pos(Q, imp),
        leafofsubconfigofzone(OMEGA, OMEGAPr, f(Q, [fst, Chi], Ls), f(Q&_, Chi, Ls)),
        p2l(Q, OMEGAPr, l(D, Omega), SubPrf).

p2l(M&_, OMEGA, l(D, Omega), prf(left(aconj), OMEGA, l(D), [SubPrf])) :-
        neg(M, imp),
        leafofsubconfigofzone(OMEGA, OMEGAPr, l(M, [fst, Chi], Ls), f(M&_, Chi, Ls)),
        p1(OMEGAPr, l(D, Omega), SubPrf).

p2l(_&Q, OMEGA, l(D, Omega), prf(left(aconj), OMEGA, l(D), [SubPrf])) :-
        pos(Q, imp),
        leafofsubconfigofzone(OMEGA, OMEGAPr, f(Q, [snd, Chi], Ls), f(_&Q, Chi, Ls)), (!),
        p2l(Q, OMEGAPr, l(D, Omega), SubPrf).

p2l(_&M, OMEGA, l(D, Omega), prf(left(aconj), OMEGA, l(D), [SubPrf])) :-
        neg(M, imp),
        leafofsubconfigofzone(OMEGA, OMEGAPr, l(M, [snd, Chi], Ls), f(_&M, Chi, Ls)), (!),
        p1(OMEGAPr, l(D, Omega), SubPrf).

% 11

p2l(V u Q, OMEGA, l(D, Omega), prf(left(univq), OMEGA, l(D), [SubPrf])) :-
        pos(Q, imp),
        leafofsubconfigofzone(OMEGA, OMEGAPr, f(QQ, [app, Chi, VV], Ls), f(V u Q, Chi, Ls)),
        tsubst(VV, V, Q, QQ), (!),
        p2l(QQ, OMEGAPr, l(D, Omega), SubPrf).

p2l(V u M, OMEGA, l(D, Omega), prf(left(univq), OMEGA, l(D), [SubPrf])) :-
        neg(M, imp),
        leafofsubconfigofzone(OMEGA, OMEGAPr, l(MM, [app, Chi, VV], Ls), f(V u M, Chi, Ls)),
        tsubst(VV, V, M, MM), (!),
        p1(OMEGAPr, l(D, Omega), SubPrf).

% 13

p2l('L'Q, OMEGA, l(D, Omega), prf(left(umod), OMEGA, l(D), [SubPrf])) :-
        pos(Q, imp),
        leafofsubconfigofzone(OMEGA, OMEGAPr, f(Q, [dn, Chi], Ls), f('L'Q, Chi, Ls)), (!),
        p2l(Q, OMEGAPr, l(D, Omega), SubPrf).

p2l('L'M, OMEGA, l(D, Omega), prf(left(umod), OMEGA, l(D), [SubPrf])) :-
        neg(M, imp),
        leafofsubconfigofzone(OMEGA, OMEGAPr, l(M, [dn, Chi], Ls), f('L'M, Chi, Ls)), (!),
        p1(OMEGAPr, l(D, Omega), SubPrf).

% 15

p2l(ab Q, OMEGA, l(D, Omega), prf(left(abrack), OMEGA, l(D), [SubPrf])) :-
        pos(Q, imp),
        subconfigofzoneandapp(OMEGA, OMEGAPr, H, Delta1, [b([]: [f(ab Q, Phi, Ls)])], Deltar),
        append(Delta1, [f(Q, Phi, Ls)|Deltar], H), (!),
        p2l(Q, OMEGAPr, l(D, Omega), SubPrf).

p2l(ab M, OMEGA, l(D, Omega), prf(left(abrack), OMEGA, l(D), [SubPrf])) :-
        neg(M, imp),
        subconfigofzoneandapp(OMEGA, OMEGAPr, H, Delta1, [b([]: [f(ab M, Phi, Ls)])], Deltar),
        append(Delta1, [l(M, Phi, Ls)|Deltar], H), (!),
        p1(OMEGAPr, l(D, Omega), SubPrf).

% 19 L

p2l(Q lca P, Zeta: Delta, l(D, Omega), prf(left(a), Zeta: Delta, l(D), [MinSubPrf, MajSubPrf])) :-
        pos(Q, imp), pos(P, out),
        partition(Zeta, StoupMin, StoupMaj),
        an(Delta, Gamma, f(Q lca P, Chi, Ls), Theta, l(P, Phi, []), l(Q, [app, Chi, Phi], Ls)),
        p2r(StoupMin: Gamma, f(P, Phi), MinSubPrf),
        p1(StoupMaj: Theta, l(D, Omega), MajSubPrf).

p2l(Q lca N, Zeta: Delta, l(D, Omega), prf(left(a), Zeta: Delta, l(D), [MinSubPrf, MajSubPrf])) :-
```

```
      pos(Q, inp), neg(N, out),
      partition(Zeta, StoupMin, StoupMaj),
      an(Delta, Gamma, f(Q lca N, Chi, Ls), Theta, l(N, Phi, []), l(Q, [app, Chi, Phi], Ls)),
      p1(StoupMin: Gamma, l(N, Phi), MinSubPrf),
      p1(StoupMaj: Theta, l(D, Omega), MajSubPrf).

p2l(M lca P, Zeta: Delta, l(D, Omega), prf(left(a), Zeta: Delta, l(D), [MinSubPrf, MajSubPrf])) :-
      neg(M, inp), pos(P, out),
      partition(Zeta, StoupMin, StoupMaj),
      an(Delta, Gamma, f(M lca P, Chi, Ls), Theta, l(P, Phi, []), l(M, [app, Chi, Phi], Ls)),
      p2r(StoupMin: Gamma, f(P, Phi), MinSubPrf),
      p1(StoupMaj: Theta, l(D, Omega), MajSubPrf).

p2l(M lca N, Zeta: Delta, l(D, Omega), prf(left(a), Zeta: Delta, l(D), [MinSubPrf, MajSubPrf])) :-
      neg(N, out), neg(M, inp),
      partition(Zeta, StoupMin, StoupMaj),
      an(Delta, Gamma, f(M lca N, Chi, Ls), Theta, l(N, Phi, []), l(M, [app, Chi, Phi], Ls)),
      p1(StoupMin: Gamma, f(N, Phi), MinSubPrf),
      p1(StoupMaj: Theta, l(D, Omega), MajSubPrf).

% 19 R

p2l(Q lca A, Stoup: Gamma, l(D lca A, [lmd, X, Psi]), prf(right(a), Stoup: Gamma, l(Q lca A), [SubPrf])) :-
      pos(Q, inp),
      abindoff(Gamma, A, X, Gamma1, foc),
      p2l(Q, Stoup: Gamma1, l(D, Psi), SubPrf).

p2l(M lca A, Stoup: Gamma, l(D lca A, [lmd, X, Psi]), prf(right(a), Stoup: Gamma, l(M lca A), [SubPrf])) :-
      neg(M, inp),
      abindoff(Gamma, A, X, Gamma1, defoc),
      p1(Stoup: Gamma1, l(D, Psi), SubPrf).

% 20

p2l(Q lio P, OMEGA, l(D, Omega), prf(left(liover), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), pos(P, out), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q lio P, _, Ls2)|Ls], Deltar),
      ssort(P, SP),
      partition(Stoup, Stoup1, Stoup2),
      fold(SP, Gamma, Ls1, Ls),
      p2r(Stoup1: Gamma, f(P, _), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [f(Q, 0, Lss)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(Q lio N, OMEGA, l(D, Omega), prf(left(liover), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), neg(N, out), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q lio N, _, Ls2)|Ls], Deltar),
      ssort(N, SN),
      partition(Stoup, Stoup1, Stoup2),
      fold(SN, Gamma, Ls1, Ls),
      p1(Stoup1: Gamma, l(N, _), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [f(Q, 0, Lss)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(M lio P, OMEGA, l(D, Omega), prf(left(liover), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(N, inp), pos(P, out), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M lio P, _, Ls2)|Ls], Deltar),
      ssort(P, SP),
      partition(Stoup, Stoup1, Stoup2),
      fold(SP, Gamma, Ls1, Ls),
      p2r(Stoup1: Gamma, f(P, _), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [l(N, 0, Lss)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(N lio M, OMEGA, l(D, Omega), prf(left(liover), OMEGA, l(D), [SubPrf1, SubPrf2])) :- (!),
      neg(N, inp), neg(M, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(N lio M, _, Ls2)|Ls], Deltar),
```

```
    ssort(M, SM),
    partition(Stoup, Stoup1, Stoup2),
    fold(SM, Gamma, Ls1, Ls),
    p1(Stoup1: Gamma, l(M, _), SubPrf1),
    append(Ls1, Ls2, Lss),
    append(Deltal, [l(N, 0, Lss)|Deltar], Delta),
    p1(OMEGAPr, l(D, Omega), SubPrf2).

% 21

p21(P liu Q, OMEGA, l(D, Omega), prf(left(liunder), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
    pos(P, out), pos(Q, inp), (!),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, Ls8, Deltar),
    append(Ls, [f(P liu Q, Psi, Ls2)], Ls8),
    ssort(P, SP),
    partition(Stoup, Stoup1, Stoup2),
    fold(SP, Gamma, Ls1, Ls),
    p2r(Stoup1: Gamma, f(P, _), SubPrf1),
    append(Ls1, Ls2, Lss),
    append(Deltal, [f(Q, Psi, Lss)|Deltar], Delta),
    p21(Q, OMEGAPr, l(D, Omega), SubPrf2).

p21(P liu N, OMEGA, l(D, Omega), prf(left(liunder), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
    pos(P, out), neg(N, inp), (!),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, Ls8, Deltar),
    append(Ls, [f(P liu N, Psi, Ls2)], Ls8),
    ssort(P, SP),
    partition(Stoup, Stoup1, Stoup2),
    fold(SP, Gamma, Ls1, Ls),
    p2r(Stoup1: Gamma, f(P, _), SubPrf1),
    append(Ls1, Ls2, Lss),
    append(Deltal, [l(N, Psi, Lss)|Deltar], Delta),
    p1(OMEGAPr, l(D, Omega), SubPrf2).

p21(N liu P, OMEGA, l(D, Omega), prf(left(liunder), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
    neg(N, out), pos(P, inp), (!),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, Ls8, Deltar),
    append(Ls, [f(N liu P, Psi, Ls2)], Ls8),
    ssort(N, SN),
    partition(Stoup, Stoup1, Stoup2),
    fold(SN, Gamma, Ls1, Ls),
    p1(Stoup1: Gamma, l(N, _), SubPrf1),
    append(Ls1, Ls2, Lss),
    append(Deltal, [f(P, Psi, Lss)|Deltar], Delta),
    p21(P, OMEGAPr, l(D, Omega), SubPrf2).

p21(N liu M, OMEGA, l(D, Omega), prf(left(liunder), OMEGA, l(D), [SubPrf1, SubPrf2])) :- (!),
    neg(N, out), neg(M, inp),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, Ls8, Deltar),
    append(Ls, [f(N liu M, Psi, Ls2)], Ls8),
    ssort(N, SN),
    partition(Stoup, Stoup1, Stoup2),
    fold(SN, Gamma, Ls1, Ls),
    p1(Stoup1: Gamma, l(N, _), SubPrf1),
    append(Ls1, Ls2, Lss),
    append(Deltal, [l(M, Psi, Lss)|Deltar], Delta),
    p1(OMEGAPr, l(D, Omega), SubPrf2).

% 22

p21(Q rio P, OMEGA, l(D, Omega), prf(left(riover), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
    pos(Q, inp), pos(P, out), (!),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q rio P, Psi, Ls2)|Ls], Deltar),
    ssort(P, SP),
    partition(Stoup, Stoup1, Stoup2),
    fold(SP, Gamma, Ls1, Ls),
    p2r(Stoup1: Gamma, f(P, _), SubPrf1),
    append(Ls1, Ls2, Lss),
    append(Deltal, [f(Q, Psi, Lss)|Deltar], Delta),
    p21(Q, OMEGAPr, l(D, Omega), SubPrf2).
```

```
p2l(Q rio N, OMEGA, l(D, Omega), prf(left(riover), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), neg(N, out), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q rio N, Psi, Ls2)|Ls], Deltar),
      ssort(N, SN),
      partition(Stoup, Stoup1, Stoup2),
      fold(SN, Gamma, Ls1, Ls),
      p1(Stoup1: Gamma, l(N, _), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [f(Q, Psi, Lss)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(M rio P, OMEGA, l(D, Omega), prf(left(riover), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(M, inp), pos(P, out), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M rio P, Psi, Ls2)|Ls], Deltar),
      ssort(P, SP),
      partition(Stoup, Stoup1, Stoup2),
      fold(SP, Gamma, Ls1, Ls),
      p2r(Stoup1: Gamma, f(P, _), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [l(M, Psi, Lss)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(N rio M, OMEGA, l(D, Omega), prf(left(riover), OMEGA, l(D), [SubPrf1, SubPrf2])) :- (!),
      neg(N, inp), neg(M, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(N rio M, Psi, Ls2)|Ls], Deltar),
      ssort(M, SM),
      partition(Stoup, Stoup1, Stoup2),
      fold(SM, Gamma, Ls1, Ls),
      p1(Stoup1: Gamma, l(M, _), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [l(N, Psi, Lss)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

% 23

p2l(P riu Q, OMEGA, l(D, Omega), prf(left(riunder), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(P, out), pos(Q, inp), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, LsB, Deltar),
      append(Ls, [f(P riu Q, _, Ls2)], LsB),
      ssort(P, SP),
      partition(Stoup, Stoup1, Stoup2),
      fold(SP, Gamma, Ls1, Ls),
      p2r(Stoup1: Gamma, f(P, _), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [f(Q, 0, Lss)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(P riu N, OMEGA, l(D, Omega), prf(left(riunder), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(P, out), neg(N, inp), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, LsB, Deltar),
      append(Ls, [f(P riu N, _, Ls2)], LsB),
      ssort(P, SP),
      partition(Stoup, Stoup1, Stoup2),
      fold(SP, Gamma, Ls1, Ls),
      p2r(Stoup1: Gamma, f(P, _), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [l(N, 0, Lss)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(N riu P, OMEGA, l(D, Omega), prf(left(riunder), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(N, out), pos(P, inp), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, LsB, Deltar),
      append(Ls, [f(N riu P, _, Ls2)], LsB),
      ssort(N, SN),
      partition(Stoup, Stoup1, Stoup2),
      fold(SN, Gamma, Ls1, Ls),
      p1(Stoup1: Gamma, l(N, _), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [f(P, 0, Lss)|Deltar], Delta),
```

```
    p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(N riu M, OMEGA, l(D, Omega), prf(left(riunder), OMEGA, l(D), [SubPrf1, SubPrf2])) :- (!),
    neg(N, out), neg(M, inp),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, LsB, Deltar),
    ssort(N, SN),
    append(Ls, [f(N riu M, _, Ls2)], LsB),
    partition(Stoup, Stoup1, Stoup2),
    fold(SN, Gamma, Ls1, Ls),
    p1(Stoup1: Gamma, l(N, _), SubPrf1),
    append(Ls1, Ls2, Lss),
    append(Deltal, [l(M, 0, Lss)|Deltar], Delta),
    p1(OMEGAPr, l(D, Omega), SubPrf2).

% 26+

p2l(Q uic P, OMEGA, l(D, Omega), prf(left(uicircum), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
    pos(Q, inp), pos(P, out),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q uic P, Phi, [Gamma|Gammas])], Deltar), (!),
    partition(Stoup, Stoup1, Stoup2),
    p2r(Stoup1: Gamma, f(P, _), SubPrf1),
    ssort(P, SP),
    do_ones(SP, Ls),
    append(Ls, Gammas, LsGammas),
    append(Deltal, [f(Q, Phi, LsGammas)|Deltar], Delta),
    p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(Q uic N, OMEGA, l(D, Omega), prf(left(uicircum), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
    pos(Q, inp), neg(N, out),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q uic N, Phi, [Gamma|Gammas])], Deltar), (!),
    partition(Stoup, Stoup1, Stoup2),
    p1(Stoup1: Gamma, l(N, _), SubPrf1),
    ssort(N, SN),
    do_ones(SN, Ls),
    append(Ls, Gammas, LsGammas),
    append(Deltal, [f(Q, Phi, LsGammas)|Deltar], Delta),
    p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(M uic P, OMEGA, l(D, Omega), prf(left(uicircum), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
    neg(M, inp), pos(P, out),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M uic P, Phi, [Gamma|Gammas])], Deltar), (!),
    partition(Stoup, Stoup1, Stoup2),
    p2r(Stoup1: Gamma, f(P, _), SubPrf1),
    ssort(P, SP),
    do_ones(SP, Ls),
    append(Ls, Gammas, LsGammas),
    append(Deltal, [l(M, Phi, LsGammas)|Deltar], Delta),
    p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(M uic N, OMEGA, l(D, Omega), prf(left(uicircum), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
    neg(M, inp), neg(N, out),
%   subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M uic N, Phi, [Gamma|Gammas])], Deltar), (!),
    partition(Stoup, Stoup1, Stoup2),
    p1(Stoup1: Gamma, l(N, _), SubPrf1),
    ssort(N, SN),
    do_ones(SN, Ls),
    append(Ls, Gammas, LsGammas),
    append(Deltal, [l(M, Phi, LsGammas)|Deltar], Delta),
    p1(OMEGAPr, l(D, Omega), SubPrf2).

% 26-

p2l(Q uicn P, OMEGA, l(D, Omega), prf(left(uicircumn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
    pos(Q, inp), pos(P, out),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q uicn P, Phi, GammasGamma)], Deltar),
    append(Gammas, [Gamma], GammasGamma), (!),
    partition(Stoup, Stoup1, Stoup2),
    p2r(Stoup1: Gamma, f(P, _), SubPrf1),
    ssort(P, SP),
    do_ones(SP, Ls),
```

```
      append(Gammas, Ls, GammasLs),
      append(Deltal, [f(Q, Phi, GammasLs)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(Q uicn N, OMEGA, l(D, Omega), prf(left(uicircumn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), neg(N, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q uicn N, Phi, GammasGamma)], Deltar),
      append(Gammas, [Gamma], GammasGamma), (!),
      partition(Stoup, Stoup1, Stoup2),
      p1(Stoup1: Gamma, l(N, _), SubPrf1),
      ssort(N, SN),
      do_ones(SN, Ls),
      append(Gammas, Ls, GammasLs),
      append(Deltal, [f(Q, Phi, GammasLs)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(M uicn P, OMEGA, l(D, Omega), prf(left(uicircumn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(M, inp), pos(P, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M uicn P, Phi, GammasGamma)], Deltar),
      append(Gammas, [Gamma], GammasGamma), (!),
      partition(Stoup, Stoup1, Stoup2),
      p2r(Stoup1: Gamma, f(P, _), SubPrf1),
      ssort(P, SP),
      do_ones(SP, Ls),
      append(Gammas, Ls, GammasLs),
      append(Deltal, [l(M, Phi, GammasLs)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(M uicn N, OMEGA, l(D, Omega), prf(left(uicircumn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(M, inp), neg(N, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M uicn N, Phi, GammasGamma)], Deltar),
      append(Gammas, [Gamma], GammasGamma), (!),
      partition(Stoup, Stoup1, Stoup2),
      p1(Stoup1: Gamma, l(N, _), SubPrf1),
      ssort(N, SN),
      do_ones(SN, Ls),
      append(Gammas, Ls, GammasLs),
      append(Deltal, [l(M, Phi, GammasLs)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

% 27+

p2l(P uii Q, OMEGA, l(D, Omega), prf(left(uiinfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(P, out), pos(Q, inp), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, [f(P uii Q, _, Ls1)|_], [], L),
      ssort(P, SP),
      partition(Stoup, Stoup1, Stoup2),
      fold(SP, Gamma2, [[1]|Ls2], Gamma1),
      p2r(Stoup1: Gamma2, f(P, _), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [f(Q, 0, Lss)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(P uii M, OMEGA, l(D, Omega), prf(left(uiinfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(P, out), neg(M, inp), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, [f(P uii M, _, Ls1)|_], [], L),
      ssort(P, SP),
      partition(Stoup, Stoup1, Stoup2),
      fold(SP, Gamma2, [[1]|Ls2], Gamma1),
      p2r(Stoup1: Gamma2, f(P, _), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [l(M, 0, Lss)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(N uii Q, OMEGA, l(D, Omega), prf(left(uiinfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(N, out), pos(Q, inp), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, [f(N uii Q, _, Ls1)|_], [], L),
```

```
    ssort(N, SN),
    partition(Stoup, Stoup1, Stoup2),
    fold(SN, Gamma2, [[1]|Ls2], Gamma1),
    p1(Stoup1: Gamma2, 1(N, _), SubPrf1),
    append(Ls1, Ls2, Ls5),
    append(Deltal, [f(Q, 0, Lss)|Deltar], Delta),
    p21(Q, OMEGAPr, 1(D, Omega), SubPrf2).

p21(N uii M, OMEGA, 1(D, Omega), prf(left(uiinfix), OMEGA, 1(D), [SubPrf1, SubPrf2])) :- (!),
        neg(N, out), neg(M, inp),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
        genins(Gamma1, [f(N uii M, Psi, Ls1)|_], [], L),
        ssort(N, SN),
        partition(Stoup, Stoup1, Stoup2),
        fold(SN, Gamma2, [[1]|Ls2], Gamma1),
        p1(Stoup1: Gamma2, 1(N, Phi), SubPrf1),
        append(Ls1, Ls2, Ls5),
        append(Deltal, [1(M, [app, Psi, Phi], Lss)|Deltar], Delta),
        p1(OMEGAPr, 1(D, Omega), SubPrf2).

% 27-

    p21(P uiin Q, OMEGA, 1(D, Omega), prf(left(uiinfixn), OMEGA, 1(D), [SubPrf1, SubPrf2])) :-
        pos(P, out), pos(Q, inp), (!),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
        genins(Gamma1, Ls, [], L),
        append(_, [f(P uiin Q, _, Ls1)], Ls),
        ssort(P, [1|N]),
        kappend(N, Ls2, [[1]], Ls3),
        partition(Stoup, Stoup1, Stoup2),
        fold([1|N], Gamma2, Ls3, Gamma1),
        p2r(Stoup1: Gamma2, f(P, _), SubPrf1),
        append(Ls2, Ls1, Lss),
        append(Deltal, [f(Q, 0, Lss)|Deltar], Delta),
        p21(Q, OMEGAPr, 1(D, Omega), SubPrf2).

p21(P uiin M, OMEGA, 1(D, Omega), prf(left(uiinfixn), OMEGA, 1(D), [SubPrf1, SubPrf2])) :-
        pos(P, out), neg(M, inp), (!),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
        genins(Gamma1, Ls, [], L),
        append(_, [f(P uiin M, _, Ls1)], Ls),
        ssort(P, [1|N]),
        kappend(N, Ls2, [[1]], Ls3),
        partition(Stoup, Stoup1, Stoup2),
        fold([1|N], Gamma2, Ls3, Gamma1),
        p2r(Stoup1: Gamma2, f(P, _), SubPrf1),
        append(Ls2, Ls1, Lss),
        append(Deltal, [1(M, 0, Lss)|Deltar], Delta),
        p1(OMEGAPr, 1(D, Omega), SubPrf2).

    p21(N uiin Q, OMEGA, 1(D, Omega), prf(left(uiinfixn), OMEGA, 1(D), [SubPrf1, SubPrf2])) :-
        neg(N, out), pos(Q, inp), (!),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
        genins(Gamma1, Ls, [], L),
        append(_, [f(N uiin Q, _, Ls1)], Ls),
        ssort(N, [1|NN]),
        kappend(NN, Ls2, [[1]], Ls3),
        partition(Stoup, Stoup1, Stoup2),
        fold([1|NN], Gamma2, Ls3, Gamma1),
        p1(Stoup1: Gamma2, 1(N, _), SubPrf1),
        append(Ls2, Ls1, Lss),
        append(Deltal, [f(Q, 0, Lss)|Deltar], Delta),
        p21(Q, OMEGAPr, 1(D, Omega), SubPrf2).

p21(N uiin M, OMEGA, 1(D, Omega), prf(left(uiinfixn), OMEGA, 1(D), [SubPrf1, SubPrf2])) :- (!),
        neg(N, out), neg(M, inp),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
        genins(Gamma1, Ls, [], L),
        append(_, [f(N uiin M, _, Ls1)], Ls),
        ssort(N, [1|NN]),
```

```
      kappend(NN, Ls2, [[1]], Ls3),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|NN], Gamma2, Ls3, Gamma1),
      p1(Stoup1: Gamma2, l(N, _), SubPrf1),
      append(Ls2, Ls1, Lss),
      append(Deltal, [l(M, 0, Lss)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

% 28+

p2l(Q lic P, OMEGA, l(D, Omega), prf(left(licircum), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), pos(P, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q lic P, Chi, [Gamma|Gammas])], Deltar), (!),
      partition(Stoup, Stoup1, Stoup2),
      p2r(Stoup1: Gamma, f(P, _), SubPrf1),
      ssort(P, SP),
      do_ones(SP, Ls),
      append(Ls, Gammas, LsGammas),
      append(Deltal, [f(Q, Chi, LsGammas)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(Q lic N, OMEGA, l(D, Omega), prf(left(licircum), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), neg(N, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q lic N, Chi, [Gamma|Gammas])], Deltar), (!),
      partition(Stoup, Stoup1, Stoup2),
      p1(Stoup1: Gamma, l(N, _), SubPrf1),
      ssort(N, SN),
      do_ones(SN, Ls),
      append(Ls, Gammas, LsGammas),
      append(Deltal, [f(Q, Chi, LsGammas)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(M lic P, OMEGA, l(D, Omega), prf(left(licircum), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(M, inp), pos(P, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M lic P, Chi, [Gamma|Gammas])], Deltar), (!),
      partition(Stoup, Stoup1, Stoup2),
      p2r(Stoup1: Gamma, f(P, _), SubPrf1),
      ssort(P, SP),
      do_ones(SP, Ls),
      append(Ls, Gammas, LsGammas),
      append(Deltal, [l(M, Chi, LsGammas)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(N lic M, OMEGA, l(D, Omega), prf(left(licircum), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(N, inp), neg(M, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(N lic M, Chi, [Gamma|Gammas])], Deltar), (!),
      partition(Stoup, Stoup1, Stoup2),
      p1(Stoup1: Gamma, l(M, _), SubPrf1),
      ssort(M, SM),
      do_ones(SM, Ls),
      append(Ls, Gammas, LsGammas),
      append(Deltal, [l(N, Chi, LsGammas)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

% 28-

p2l(Q licn P, OMEGA, l(D, Omega), prf(left(licircumn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), pos(P, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q licn P, Chi, GammasGamma)], Deltar),
      append(Gammas, [Gamma], GammasGamma), (!),
      partition(Stoup, Stoup1, Stoup2),
      p2r(Stoup1: Gamma, f(P, _), SubPrf1),
      ssort(P, SP),
      do_ones(SP, Ls),
      append(Gammas, Ls, GammasLs),
      append(Deltal, [f(Q, Chi, GammasLs)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(Q licn N, OMEGA, l(D, Omega), prf(left(licircumn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), neg(N, out),
```

```
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q licn N, Chi, GammasGamma)], Deltar),
    append(Gammas, [Gamma], GammasGamma),
    partition(Stoup, Stoup1, Stoup2),
    p1(Stoup1: Gamma, 1(N, _), SubPrf1),
    ssort(N, SN),
    do_ones(SN, Ls),
    append(Gammas, Ls, GammasLs),
    append(Deltal, [f(Q, Chi, GammasLs)|Deltar], Delta),
    p2l(Q, OMEGAPr, 1(D, Omega), SubPrf2).

p2l(M licn P, OMEGA, 1(D, Omega), prf(left(licircumn), OMEGA, 1(D), [SubPrf1, SubPrf2])) :-
    neg(M, inp), pos(P, out),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M licn P, Chi, GammasGamma)], Deltar),
    append(Gammas, [Gamma], GammasGamma), (!),
    partition(Stoup, Stoup1, Stoup2),
    p2r(Stoup1: Gamma, f(P, _), SubPrf1),
    ssort(P, SP),
    do_ones(SP, Ls),
    append(Gammas, Ls, GammasLs),
    append(Deltal, [1(M, Chi, GammasLs)|Deltar], Delta),
    p1(OMEGAPr, 1(D, Omega), SubPrf2).

p2l(N licn M, OMEGA, 1(D, Omega), prf(left(licircumn), OMEGA, 1(D), [SubPrf1, SubPrf2])) :-
    neg(N, inp), neg(M, out),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(N licn M, Chi, GammasGamma)], Deltar),
    append(Gammas, [Gamma], GammasGamma),
    partition(Stoup, Stoup1, Stoup2),
    p1(Stoup1: Gamma, 1(M, _), SubPrf1),
    ssort(M, SM),
    do_ones(SM, Ls),
    append(Gammas, Ls, GammasLs),
    append(Deltal, [1(N, Chi, GammasLs)|Deltar], Delta),
    p1(OMEGAPr, 1(D, Omega), SubPrf2).

% 29+

p2l(P lii Q, OMEGA, 1(D, Omega), prf(left(liinfix), OMEGA, 1(D), [SubPrf1, SubPrf2])) :-
    pos(P, out), pos(Q, inp), (!),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
    genins(Gamma1, [f(P lii Q, _, Ls1)|_], [], L),
    ssort(P, SP),
    partition(Stoup, Stoup1, Stoup2),
    fold(SP, Gamma2, [[1]|Ls2], Gamma1),
    p2r(Stoup1: Gamma2, f(P, _), SubPrf1),
    append(Ls1, Ls2, Ls5),
    append(Deltal, [f(Q, 0, Ls5)|Deltar], Delta),
    p2l(Q, OMEGAPr, 1(D, Omega), SubPrf2).

p2l(P lii M, OMEGA, 1(D, Omega), prf(left(liinfix), OMEGA, 1(D), [SubPrf1, SubPrf2])) :-
    pos(P, out), neg(M, inp), (!),
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
    genins(Gamma1, [f(P lii M, _, Ls1)|_], [], L),
    ssort(P, SP),
    partition(Stoup, Stoup1, Stoup2),
    fold(SP, Gamma2, [[1]|Ls2], Gamma1),
    p2r(Stoup1: Gamma2, f(P, _), SubPrf1),
    append(Ls1, Ls2, Ls5),
    append(Deltal, [1(M, 0, Ls5)|Deltar], Delta),
    p1(OMEGAPr, 1(D, Omega), SubPrf2).

p2l(N lii Q, OMEGA, 1(D, Omega), prf(left(liinfix), OMEGA, 1(D), [SubPrf1, SubPrf2])) :-
    subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
    genins(Gamma1, [f(N lii Q, _, Ls1)|_], [], L),
    neg(N, out), pos(Q, inp),
    ssort(N, SN),
    partition(Stoup, Stoup1, Stoup2),
    fold(SN, Gamma2, [[1]|Ls2], Gamma1),
    p1(Stoup1: Gamma2, 1(N, _), SubPrf1),
    append(Ls1, Ls2, Ls5),
    append(Deltal, [f(Q, 0, Ls5)|Deltar], Delta),
```

```
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(N lii M, OMEGA, l(D, Omega), prf(left(liinfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :- (!),
      neg(N, out), neg(M, inp),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, [f(N lii M, _, Ls1)|_], [], L),
      ssort(N, SN),
      partition(Stoup, Stoup1, Stoup2),
      fold(SN, Gamma2, [[1]|Ls2], Gamma1),
      p1(Stoup1: Gamma2, l(N, _), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [l(M, 0, Lss)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

% 29-

p2l(P liin Q, OMEGA, l(D, Omega), prf(left(liinfixn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(P, out), pos(Q, inp), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, Ls, [], L),
      append(_, [f(P liin Q, _, Ls1)], Ls),
      ssort(P, [1|N]),
      kappend(N, Ls2, [[1]], Ls3),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|N], Gamma2, Ls3, Gamma1),
      p2r(Stoup1: Gamma2, f(P, _), SubPrf1),
      append(Ls2, Ls1, Lss),
      append(Deltal, [f(Q, 0, Lss)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(P liin M, OMEGA, l(D, Omega), prf(left(liinfixn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(P, out), neg(M, inp), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, Ls, [], L),
      append(_, [f(P liin M, _, Ls1)], Ls),
      ssort(P, [1|N]),
      kappend(N, Ls2, [[1]], Ls3),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|N], Gamma2, Ls3, Gamma1),
      p2r(Stoup1: Gamma2, f(P, _), SubPrf1),
      append(Ls2, Ls1, Lss),
      append(Deltal, [l(M, 0, Lss)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(N liin Q, OMEGA, l(D, Omega), prf(left(liinfixn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(N, out), pos(Q, inp), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, Ls, [], L),
      append(_, [f(N liin Q, _, Ls1)], Ls),
      ssort(N, [1|NN]),
      kappend(NN, Ls2, [[1]], Ls3),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|NN], Gamma2, Ls3, Gamma1),
      p1(Stoup1: Gamma2, l(N, _), SubPrf1),
      append(Ls2, Ls1, Lss),
      append(Deltal, [f(Q, 0, Lss)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(N liin M, OMEGA, l(D, Omega), prf(left(liinfixn), OMEGA, l(D), [SubPrf1, SubPrf2])) :- (!),
      neg(N, out), neg(M, inp),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, Ls, [], L),
      append(_, [f(N liin M, _, Ls1)], Ls),
      ssort(N, [1|NN]),
      kappend(NN, Ls2, [[1]], Ls3),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|NN], Gamma2, Ls3, Gamma1),
      p1(Stoup1: Gamma2, l(N, _), SubPrf1),
      append(Ls2, Ls1, Lss),
      append(Deltal, [l(M, 0, Lss)|Deltar], Delta),
```

```
    p1(OMEGAPr, l(D, Omega), SubPrf2).

% 33

p2l(Q iac _, OMEGA, l(D, Omega), prf(left(iaconj), OMEGA, l(D), [SubPrf])) :-
    pos(Q, inp),
    leafofsubconfigofzone(OMEGA, OMEGAPr, f(Q, Chi, Ls), f(Q iac _, Chi, Ls)),
    p2l(Q, OMEGAPr, l(D, Omega), SubPrf).

p2l(M iac _, OMEGA, l(D, Omega), prf(left(iaconj), OMEGA, l(D), [SubPrf])) :-
    neg(M, inp),
    leafofsubconfigofzone(OMEGA, OMEGAPr, l(M, Chi, Ls), f(M iac _, Chi, Ls)),
    p1(OMEGAPr, l(D, Omega), SubPrf).

p2l(_ iac Q, OMEGA, l(D, Omega), prf(left(iaconj), OMEGA, l(D), [SubPrf])) :-
    pos(Q, inp),
    leafofsubconfigofzone(OMEGA, OMEGAPr, f(Q, Chi, Ls), f(_ iac Q, Chi, Ls)), (!),
    p2l(Q, OMEGAPr, l(D, Omega), SubPrf).

p2l(_ iac M, OMEGA, l(D, Omega), prf(left(iaconj), OMEGA, l(D), [SubPrf])) :-
    neg(M, inp),
    leafofsubconfigofzone(OMEGA, OMEGAPr, l(M, Chi, Ls), f(_ iac M, Chi, Ls)), (!),
    p1(OMEGAPr, l(D, Omega), SubPrf).

% 35

p2l(V iu Q, OMEGA, l(D, Omega), prf(left(iunivq), OMEGA, l(D), [SubPrf])) :-
    pos(Q, inp),
    leafofsubconfigofzone(OMEGA, OMEGAPr, f(QQ, Chi, Ls), f(V iu Q, Chi, Ls)),
    tsubst(_, V, Q, QQ), (!),
    p2l(QQ, OMEGAPr, l(D, Omega), SubPrf).

p2l(V iu M, OMEGA, l(D, Omega), prf(left(iunivq), OMEGA, l(D), [SubPrf])) :-
    neg(M, inp),
    leafofsubconfigofzone(OMEGA, OMEGAPr, l(MM, Chi, Ls), f(V iu M, Chi, Ls)),
    tsubst(_, V, M, MM), (!),
    p1(OMEGAPr, l(D, Omega), SubPrf).

% 37

p2l(iL Q, OMEGA, l(D, Omega), prf(left(iumod), OMEGA, l(D), [SubPrf])) :-
    pos(Q, inp),
    leafofsubconfigofzone(OMEGA, OMEGAPr, f(Q, Chi, Ls), f(iL Q, Chi, Ls)), (!),
    p2l(Q, OMEGAPr, l(D, Omega), SubPrf).

p2l(iL M, OMEGA, l(D, Omega), prf(left(iumod), OMEGA, l(D), [SubPrf])) :-
    neg(M, inp),
    leafofsubconfigofzone(OMEGA, OMEGAPr, l(M, Chi, Ls), f(iL M, Chi, Ls)), (!),
    p1(OMEGAPr, l(D, Omega), SubPrf).

% 39

p2l(lp Q, OMEGA, l(B, Psi), prf(left(lproj), OMEGA, l(B), [SubPrf])) :-
    pos(Q, inp),
    subconfigofzoneandapp(OMEGA, OMEGAPr, H, Delta1, [f(lp Q, Phi, Ls), 1], Deltar),
    append(Delta1, [f(Q, Phi, Ls)|Delta1], H), (!),
    p2l(Q, OMEGAPr, l(B, Psi), SubPrf).

p2l(lp M, OMEGA, l(B, Psi), prf(left(lproj), OMEGA, l(B), [SubPrf])) :-
    neg(M, inp),
    subconfigofzoneandapp(OMEGA, OMEGAPr, H, Delta1, [f(lp M, Phi, Ls), 1], Deltar),
    append(Delta1, [l(M, Phi, Ls)|Delta1], H), (!),
    p1(OMEGAPr, l(B, Psi), SubPrf).

% 40

p2l(rp Q, OMEGA, l(B, Psi), prf(left(rproj), OMEGA, l(B), [SubPrf])) :-
    pos(Q, inp),
    subconfigofzoneandapp(OMEGA, OMEGAPr, H, Delta1, [1, f(rp Q, Phi, Ls)], Deltar),
    append(Delta1, [f(Q, Phi, Ls)|Delta1], H), (!),
```

```
      p2l(Q, OMEGAPr, l(B, Psi), SubPrf).

p2l(rp M, OMEGA, l(B, Psi), prf(left(rproj), OMEGA, l(B), [SubPrf])) :-
      neg(M, inp),
      subconfigofzoneandapp(OMEGA, OMEGAPr, H, Deltal, [1, f(rp M, Phi, Ls)], Deltar),
      append(Deltal, [l(M, Phi, Ls)|Deltar], H), (!),
      p1(OMEGAPr, l(B, Psi), SubPrf).

% 43+

p2l(sp Q, OMEGA, l(C, Chi), prf(left(split), OMEGA, l(C), [SubPrf])) :-
      pos(Q, inp),
      leafofsubconfigofzone(OMEGA, OMEGAPr, f(Q, Psi, Ls), f(sp Q, Psi, [[]|Ls])), (!),
      p2l(Q, OMEGAPr, l(C, Chi), SubPrf).

p2l(sp M, OMEGA, l(C, Chi), prf(left(split), OMEGA, l(C), [SubPrf])) :-
      neg(M, inp),
      leafofsubconfigofzone(OMEGA, OMEGAPr, l(M, Psi, Ls), f(sp M, Psi, [[]|Ls])), (!),
      p1(OMEGAPr, l(C, Chi), SubPrf).

% 43-

p2l(spn Q, OMEGA, l(C, Chi), prf(left(splitn), OMEGA, l(C), [SubPrf])) :-
      pos(Q, inp),
      leafofsubconfigofzone(OMEGA, OMEGAPr, f(Q, Psi, Ls), f(sp Q, Psi, LsLambda)),
      append(Ls, [[]], LsLambda), (!),
      p2l(Q, OMEGAPr, l(C, Chi), SubPrf).

p2l(spn M, OMEGA, l(C, Chi), prf(left(splitn), OMEGA, l(C), [SubPrf])) :-
      neg(M, inp),
      leafofsubconfigofzone(OMEGA, OMEGAPr, l(M, Psi, Ls), f(sp M, Psi, LsLambda)),
      append(Ls, [[]], LsLambda), (!),
      p1(OMEGAPr, l(C, Chi), SubPrf).

% 45

p2l(Q nd P, OMEGA, l(D, Omega), prf(left(ndiv), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), pos(P, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, LsB, Deltar),
      append(Ls, [f(Q nd P, Psi, Ls2)], LsB),
      ssort(P, SP),
      partition(Stoup, Stoup1, Stoup2),
      fold(SP, Gamma, Ls1, Ls),
      p2r(Stoup1: Gamma, f(P, Phi), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [f(Q, [app, Psi, Phi], Lss)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(Q nd N, OMEGA, l(D, Omega), prf(left(ndiv), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), neg(N, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, LsB, Deltar),
      append(Ls, [f(Q nd N, Psi, Ls2)], LsB),
      ssort(N, SN),
      partition(Stoup, Stoup1, Stoup2),
      fold(SN, Gamma, Ls1, Ls),
      p1(Stoup1: Gamma, l(N, Phi), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [l(Q, [app, Psi, Phi], Lss)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(M nd P, OMEGA, l(D, Omega), prf(left(ndiv), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(M, inp), pos(P, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, LsB, Deltar),
      append(Ls, [f(M nd P, Psi, Ls2)], LsB),
      ssort(P, SP),
      partition(Stoup, Stoup1, Stoup2),
      fold(SP, Gamma, Ls1, Ls),
      p2r(Stoup1: Gamma, f(P, Phi), SubPrf1),
      append(Ls1, Ls2, Lss),
      append(Deltal, [l(M, [app, Psi, Phi], Lss)|Deltar], Delta),
```

```
    p1(OMEGAPr, 1(D, Omega), SubPrf2).

p2l(M nd N, OMEGA, 1(D, Omega), prf(left(ndiv), OMEGA, 1(D), [SubPrf1, SubPrf2])) :-
        neg(M, inp), neg(N, out),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, LsB, Deltar),
        ssort(N, SN),
        partition(Stoup, Stoup1, Stoup2),
        fold(SN, Gamma, Ls1, Ls),
        p1(Stoup1: Gamma, 1(N, Phi)), SubPrf1),
        append(Ls1, Ls2, Lss),
        append(Deltal, [1(M, [app, Psi, Phi], Lss)|Deltar], Delta),
        p1(OMEGAPr, 1(D, Omega), SubPrf2).

p2l(Q nd P, OMEGA, 1(D, Omega), prf(left(ndiv), OMEGA, 1(D), [SubPrf1, SubPrf2])) :-
        pos(Q, inp), pos(P, out), (!),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q nd P, Psi, Ls2)|Ls], Deltar),
        ssort(P, SP),
        partition(Stoup, Stoup1, Stoup2),
        fold(SP, Gamma, Ls1, Ls),
        p2r(Stoup1: Gamma, f(P, Phi), SubPrf1),
        append(Ls1, Ls2, Lss),
        append(Deltal, [f(Q, [app, Psi, Phi], Lss)|Deltar], Delta),
        p2l(Q, OMEGAPr, 1(D, Omega), SubPrf2).

p2l(Q nd N, OMEGA, 1(D, Omega), prf(left(ndiv), OMEGA, 1(D), [SubPrf1, SubPrf2])) :-
        pos(Q, inp), neg(N, out), (!),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q nd N, Psi, Ls2)|Ls], Deltar),
        ssort(N, SN),
        partition(Stoup, Stoup1, Stoup2),
        fold(SN, Gamma, Ls1, Ls),
        p1(Stoup1: Gamma, 1(N, Phi)), SubPrf1),
        append(Ls1, Ls2, Lss),
        append(Deltal, [f(Q, [app, Psi, Phi], Lss)|Deltar], Delta),
        p2l(Q, OMEGAPr, 1(D, Omega), SubPrf2).

p2l(M nd P, OMEGA, 1(D, Omega), prf(left(ndiv), OMEGA, 1(D), [SubPrf1, SubPrf2])) :-
        neg(M, inp), pos(P, out), (!),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M nd P, Psi, Ls2)|Ls], Deltar),
        ssort(P, SP),
        partition(Stoup, Stoup1, Stoup2),
        fold(SP, Gamma, Ls1, Ls),
        p2r(Stoup1: Gamma, f(P, Phi), SubPrf1),
        append(Ls1, Ls2, Lss),
        append(Deltal, [1(M, [app, Psi, Phi], Lss)|Deltar], Delta),
        p1(OMEGAPr, 1(D, Omega), SubPrf2).

p2l(M nd N, OMEGA, 1(D, Omega), prf(left(ndiv), OMEGA, 1(D), [SubPrf1, SubPrf2])) :- (!),
        neg(N, inp), neg(M, out),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M nd N, Psi, Ls2)|Ls], Deltar),
        ssort(N, SN),
        partition(Stoup, Stoup1, Stoup2),
        fold(SN, Gamma, Ls1, Ls),
        p1(Stoup1: Gamma, 1(N, Phi)), SubPrf1),
        append(Ls1, Ls2, Lss),
        append(Deltal, [1(M, [app, Psi, Phi], Lss)|Deltar], Delta),
        p1(OMEGAPr, 1(D, Omega), SubPrf2).

% 47

p2l(Q nci P, OMEGA, 1(D, Omega), prf(left(ncircumfix), OMEGA, 1(D), [SubPrf1, SubPrf2])) :-
        pos(Q, inp), pos(P, out),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q nci P, Phi, [Gamma|Gammas])], Deltar),
        partition(Stoup, Stoup1, Stoup2),
        p2r(Stoup1: Gamma, f(P, Psi), SubPrf1),
        ssort(P, SP),
        do_ones(SP, Ls),
        append(Ls, Gammas, LsGammas),
        append(Deltal, [f(Q, [app, Phi, Psi], LsGammas)|Deltar], Delta),
        p2l(Q, OMEGAPr, 1(D, Omega), SubPrf2).
```

```
p2l(Q nci N, OMEGA, l(D, Omega), prf(left(ncircumfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), neg(N, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q nci N, Phi, [Gamma|Gammas])], Deltar),
      partition(Stoup, Stoup1, Stoup2),
      p1(Stoup1: Gamma, l(N, Psi), SubPrf1),
      ssort(N, SN),
      do_ones(SN, Ls),
      append(Ls, Gammas, LsGammas),
      append(Deltal, [f(Q, [app, Phi, Psi], LsGammas)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(M nci P, OMEGA, l(D, Omega), prf(left(ncircumfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(M, inp), pos(P, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M nci P, Phi, [Gamma|Gammas])], Deltar),
      partition(Stoup, Stoup1, Stoup2),
      p2r(Stoup1: Gamma, f(P, Psi), SubPrf1),
      ssort(P, SP),
      do_ones(SP, Ls),
      append(Ls, Gammas, LsGammas),
      append(Deltal, [l(M, [app, Phi, Psi], LsGammas)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(M nci N, OMEGA, l(D, Omega), prf(left(ncircumfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(M, inp), neg(N, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M nci N, Phi, [Gamma|Gammas])], Deltar),
      partition(Stoup, Stoup1, Stoup2),
      p1(Stoup1: Gamma, l(N, Psi), SubPrf1),
      ssort(N, SN),
      do_ones(SN, Ls),
      append(Ls, Gammas, LsGammas),
      append(Deltal, [l(M, [app, Phi, Psi], LsGammas)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(Q nci P, OMEGA, l(D, Omega), prf(left(ncircumfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), pos(P, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q nci P, Phi, GammasGamma)], Deltar),
      append(Gammas, [Gamma], GammasGamma), (!),
      partition(Stoup, Stoup1, Stoup2),
      p2r(Stoup1: Gamma, f(P, Psi), SubPrf1),
      ssort(P, SP),
      do_ones(SP, Ls),
      append(Gammas, Ls, GammasLs),
      append(Deltal, [f(Q, [app, Phi, Psi], GammasLs)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(Q nci N, OMEGA, l(D, Omega), prf(left(ncircumfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(Q, inp), neg(N, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(Q nci N, Phi, GammasGamma)], Deltar),
      append(Gammas, [Gamma], GammasGamma), (!),
      partition(Stoup, Stoup1, Stoup2),
      p1(Stoup1: Gamma, l(N, Psi), SubPrf1),
      ssort(N, SN),
      do_ones(SN, Ls),
      append(Gammas, Ls, GammasLs),
      append(Deltal, [f(Q, [app, Phi, Psi], GammasLs)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(M nci P, OMEGA, l(D, Omega), prf(left(ncircumfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(M, inp), pos(P, out),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M nci P, Phi, GammasGamma)], Deltar),
      append(Gammas, [Gamma], GammasGamma), (!),
      partition(Stoup, Stoup1, Stoup2),
      p2r(Stoup1: Gamma, f(P, Psi), SubPrf1),
      ssort(P, SP),
      do_ones(SP, Ls),
      append(Gammas, Ls, GammasLs),
      append(Deltal, [l(M, [app, Phi, Psi], GammasLs)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).
```

```
p2l(M nci N, OMEGA, l(D, Omega), prf(left(circumfixn), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
        neg(M, inp), neg(N, out),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, [f(M nci N, Phi, GammasGamma)], Deltar),
        append(Gammas, [Gamma], GammasGamma), (!),
        partition(Stoup, Stoup1, Stoup2),
        p1(Stoup1: Gamma, l(N, Psi), SubPrf1),
        ssort(N, SN),
        do_ones(SN, Ls),
        append(Gammas, Ls, GammasLs),
        append(Deltal, [l(M, [app, Phi, Psi], GammasLs)|Deltar], Delta),
        p1(OMEGAPr, l(D, Omega), SubPrf2).

    % 48

    p2l(P nin Q, OMEGA, l(D, Omega), prf(left(ninfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
        pos(P, out), pos(Q, inp),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
        genins(Gamma1, [f(P nin Q, Psi, Ls1)|_], [], L),
        ssort(P, SP),
        partition(Stoup, Stoup1, Stoup2),
        fold(SP, Gamma2, [[1]|Ls2], Gamma1),
        p2r(Stoup1: Gamma2, f(P, Phi), SubPrf1),
        append(Ls1, Ls2, Lss),
        append(Deltal, [f(Q, [app, Psi, Phi], Lss)|Deltar], Delta),
        p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

    p2l(P nin M, OMEGA, l(D, Omega), prf(left(ninfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
        pos(P, out), neg(M, inp),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
        genins(Gamma1, [f(P nin M, Psi, Ls1)|_], [], L),
        ssort(P, SP),
        partition(Stoup, Stoup1, Stoup2),
        fold(SP, Gamma2, [[1]|Ls2], Gamma1),
        p2r(Stoup1: Gamma2, f(P, Phi), SubPrf1),
        append(Ls1, Ls2, Lss),
        append(Deltal, [l(M, [app, Psi, Phi], Lss)|Deltar], Delta),
        p1(OMEGAPr, l(D, Omega), SubPrf2).

    p2l(N nin Q, OMEGA, l(D, Omega), prf(left(ninfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
        neg(N, out), pos(Q, inp),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
        genins(Gamma1, [f(N nin Q, Psi, Ls1)|_], [], L),
        ssort(N, SN),
        partition(Stoup, Stoup1, Stoup2),
        fold(SN, Gamma2, [[1]|Ls2], Gamma1),
        p1(Stoup1: Gamma2, l(N, Phi), SubPrf1),
        append(Ls1, Ls2, Lss),
        append(Deltal, [f(Q, [app, Psi, Phi], Lss)|Deltar], Delta),
        p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

    p2l(N nin M, OMEGA, l(D, Omega), prf(left(ninfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
        neg(N, out), neg(M, inp),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
        genins(Gamma1, [f(N nin M, Psi, Ls1)|_], [], L),
        ssort(N, SN),
        partition(Stoup, Stoup1, Stoup2),
        fold(SN, Gamma2, [[1]|Ls2], Gamma1),
        p1(Stoup1: Gamma2, l(N, Phi), SubPrf1),
        append(Ls1, Ls2, Lss),
        append(Deltal, [l(M, [app, Psi, Phi], Lss)|Deltar], Delta),
        p1(OMEGAPr, l(D, Omega), SubPrf2).

    p2l(P nin Q, OMEGA, l(D, Omega), prf(left(ninfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
        pos(P, out), pos(Q, inp), (!),
        subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
        genins(Gamma1, Ls, [], L),
        append(_, [f(P nin Q, Psi, Ls1)], Ls),
        ssort(P, [1|N]),
        kappend(N, Ls2, [[1]], Ls3),
        partition(Stoup, Stoup1, Stoup2),
```

```
      fold([1|N], Gamma2, Ls3, Gamma1),
      p2r(Stoup1: Gamma2, f(P, Phi), SubPrf1),
      append(Ls2, Ls1, Lss),
      append(Deltal, [f(Q, [app, Psi, Phi], Lss)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(P nin M, OMEGA, l(D, Omega), prf(left(ninfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      pos(P, out), neg(M, inp), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, Ls, [], L),
      append(_, [f(P nin M, Psi, Ls1)], Ls),
      ssort(P, [1|N]),
      kappend(N, Ls2, [[1]], Ls3),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|N], Gamma2, Ls3, Gamma1),
      p2r(Stoup1: Gamma2, f(P, Phi), SubPrf1),
      append(Ls2, Ls1, Lss),
      append(Deltal, [l(M, [app, Psi, Phi], Lss)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

p2l(N nin Q, OMEGA, l(D, Omega), prf(left(ninfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :-
      neg(N, out), pos(Q, inp), (!),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, Ls, [], L),
      append(_, [f(N nin Q, Psi, Ls1)], Ls),
      ssort(N, [1|NN]),
      kappend(NN, Ls2, [[1]], Ls3),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|NN], Gamma2, Ls3, Gamma1),
      p1(Stoup1: Gamma2, l(N, Phi), SubPrf1),
      append(Ls2, Ls1, Lss),
      append(Deltal, [f(Q, [app, Psi, Phi], Lss)|Deltar], Delta),
      p2l(Q, OMEGAPr, l(D, Omega), SubPrf2).

p2l(N nin M, OMEGA, l(D, Omega), prf(left(ninfix), OMEGA, l(D), [SubPrf1, SubPrf2])) :- (!),
      neg(N, out), neg(M, inp),
      subzoneandapp(OMEGA, OMEGAPr, Stoup2: Delta, Stoup, Deltal, L, Deltar),
      genins(Gamma1, Ls, [], L),
      append(_, [f(N nin M, Psi, Ls1)], Ls),
      ssort(N, [1|NN]),
      kappend(NN, Ls2, [[1]], Ls3),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|NN], Gamma2, Ls3, Gamma1),
      p1(Stoup1: Gamma2, l(N, Phi), SubPrf1),
      append(Ls2, Ls1, Lss),
      append(Deltal, [l(M, [app, Psi, Phi], Lss)|Deltar], Delta),
      p1(OMEGAPr, l(D, Omega), SubPrf2).

% an(+Delta, -Antec, -Anaph, -Theta, -Antec2, -Anaph2) means that configuration Delta
% contains the subconfiguration Antec followed later by type Anaph and that
% Theta is the result of replacing Antec and Anaph in Delta by variables Antec2 and
% Anaph2 respectively.

an(Delta, Antec, Anaph, Theta, Antec2, Anaph2) :-
      an2(Delta, [Antec, Anaph], [], Theta, [Antec2, Anaph2], []).

an2([], An, An, [], An2, An2).

an2(Gamma, [Gamma1, Anaph], An, [Antec2|Theta], [Antec2, Anaph2], An2) :-
      append(Gamma1, Gamma2, Gamma),
      an2(Gamma2, [Anaph], An, Theta, [Anaph2], An2).

an2([Anaph|Gamma], [Anaph], [], [Anaph2|Gamma], [Anaph2], []) :- (!).

an2([b(Stoup: Gamma)|Gamma2], AnIn, AnOut, [b(Stoup: Theta)|Theta2], AnIn2, AnOut2) :-
      an2(Gamma, AnIn, AnPr, Theta, AnIn2, AnPr2),
      an2(Gamma2, AnPr, AnOut, Theta2, AnPr2, AnOut2).

an2([l(A, Phi, Int)|Gamma], AnIn, AnOut, [l(A, Phi, Int2)|Gamma2], AnIn2, AnOut2) :-
      anlst(Int, AnIn, AnPr, Int2, AnIn2, AnPr2),
```

```
    an2(Gamma, AnPr, AnOut, Gamma2, AnPr2, AnOut2).

an2([1|Gamma], AnIn, AnOut, [1|Theta], AnIn2, AnOut2) :-
    an2(Gamma, AnIn, AnOut, Theta, AnIn2, AnOut2).

anlst([], An, An, [], An2, An2).

anlst([H|T], AnIn, AnOut, [H2|T2], AnIn2, AnOut2) :-
    an2(H, AnIn, AnPr, H2, AnIn2, AnPr2),
    anlst(T, AnPr, AnOut, T2, AnPr2, AnOut2).

% Synchronous stoup rules 17

p2s(OMEGA, 1(D, Omega), prf(perm(uexp), OMEGA, 1(D), [SubPrf])) :-
    subzone(OMEGA, OMEGAPr, StoupPr: DeltaPr, Stoup: Delta),
    append(Stoup1, [f(A, Phi, [])|Stoup2], Stoup),
%       pos(N,inp),
    append(Stoup1, Stoup2, StoupPr),
    append(Delta1, Delta2, Delta),
    append(Delta1, [1(A, Phi, [])|Delta2], DeltaPr),
    p1(OMEGAPr, 1(D, Omega), SubPrf).

p2s(OMEGA, 1(D, Omega), prf(contr(uexp), OMEGA, 1(D), [SubPrf])) :-
    subzone(OMEGA, OMEGAPr, Stoup: Delta, Stoup: Delta1GammaDelta2),
    append(_, [f(A, Phi, [])|_], Stoup),
    append(Delta1, [b([]: [b([] : Gamma)])|Delta2], Delta1GammaDelta2),
    append(Delta1, [b([1(A, Phi, [])]: Gamma)|Delta2], Delta),
    p2s(OMEGAPr, 1(D, Omega), SubPrf).

neg(_, out).          % 1
neg(_ bs _, out).     % 2
neg(_ *_, inp).       % 3
neg(i, inp).          % 4

neg(_ ci _, out).     % 5+
neg(_ cin _, out).    % 5-
neg(_ in _, out).     % 6+
neg(_ inn _, out).    % 6-
neg(_ dp _, inp).     % 7+
neg(_ dpn _, inp).    % 7-
neg(j, inp).          % 8

neg(_ & _, out).      % 9
neg(_ + _, inp).      % 10

neg(_ u _, out).      % 11
neg(_ e _, inp).      % 12

neg('L' _, out).      % 13
neg('M' _, inp).      % 14

neg(ab _, out).       % 15
neg(br _, inp).       % 16

neg('!' _, inp).      % 17
neg('!' _, out).      % 17
neg('?' _, inp).      % 18

neg(_ lca _, out).    % 19

neg(_ lio _, out).    % 20
neg(_ liu _, out).    % 21
neg(_ rio _, out).    % 22
neg(_ riu _, out).    % 23
neg(_ lip _, inp).    % 24
neg(_ rip _, inp).    % 25

neg(_ uic _, out).    % 26+
neg(_ uicn _, out).   % 26-
neg(_ uii _, out).    % 27+
```

```
neg(_ uiin _, out).  % 27-
neg(_ lic _, out).   % 28+
neg(_ licn _, out).  % 28-
neg(_ lii _, out).   % 29+
neg(_ liin _, out).  % 29-
neg(_ uidp _, inp).  % 30+
neg(_ uidpn _, inp). % 30-
neg(_ lidp _, inp).  % 31+
neg(_ lidpn _, inp). % 31-

%neg(w(_), inp).      % 32

neg(_ iac _, out).   % 33
neg(_ iad _, inp).   % 34

neg(_ iu _, out).    % 35
neg(_ ie _, inp).    % 36

neg(iL _, out).      % 37
neg(iM _, inp).      % 38

neg(lp _, out).      % 39
neg(rp _, out).      % 40
neg(li _, inp).      % 41
neg(ri _, inp).      % 42

neg(sp _, out).      % 43+
neg(spn _, out).     % 43-
neg(bg _, inp).      % 44+
neg(bgn _, inp).     % 44-

neg(_ nd _, out).    % 45
neg(_ np _, inp).    % 46

neg(_ nci _, out).   % 47
neg(_ nin _, out).   % 48
neg(_ ndp _, inp).   % 49

neg(A, out) :- primitive(A, _), atfoc(inp).
neg(A, inp) :- primitive(A, _), atfoc(out).

pos(_/_, inp).       % 1
pos(_ bs _, inp).    % 2
pos(_"_, out).       % 3
pos(i, out).         % 4

pos(_ ci _, inp).    % 5+
pos(_ cin _, inp).   % 5-
pos(_ in _, inp).    % 6+
pos(_ inn _, inp).   % 6-
pos(_ dp _, out).    % 7+
pos(_ dpn _, out).   % 7-
pos(j, out).         % 8

pos(_ & _, inp).     % 9
pos(_ + _, out).     % 10

pos(_ u _, inp).     % 11
pos(_ e _, out).     % 12

pos('L' _, inp).     % 13
pos('M' _, out).     % 14

pos(ab _, inp).      % 15
pos(br _, out).      % 16

%pos('!' _, inp).     % 17
pos('?' _, out).     % 18

pos(_ lca _, inp).   % 19
```

```
pos(_ lio _, inp).   % 20
pos(_ liu _, inp).   % 21
pos(_ rio _, inp).   % 22
pos(_ riu _, inp).   % 23
pos(_ lip _, out).   % 24
pos(_ rip _, out).   % 25

pos(_ uic _, inp).   % 26+
pos(_ uicn _, inp).  % 26-
pos(_ uii _, inp).   % 27+
pos(_ uiin _, inp).  % 27-
pos(_ lic _, inp).   % 28+
pos(_ licn _, inp).  % 28-
pos(_ lii _, inp).   % 29+
pos(_ liin _, inp).  % 29-

pos(_ uidp _, out).  % 30+
pos(_ uidpn _, out). % 30-
pos(_ lidp _, out).  % 31+
pos(_ lidpn _, out). % 31-

%pos(w(_), out).     % 32

pos(_ iac _, inp).   % 33
pos(_ iad _, out).   % 34

pos(_ iu _, inp).    % 35
pos(_ ie _, out).    % 36

pos(iL _, inp).      % 37
pos(iM _, out).      % 38

pos(lp _, inp).      % 39
pos(rp _, inp).      % 40
pos(li _, out).      % 41
pos(ri _, out).      % 42

pos(sp _, inp).      % 43+
pos(spn _, inp).     % 43-
pos(bg _, out).      % 44+
pos(bgn _, out).     % 44-

pos(_ nd _, inp).    % 45
pos(_ np _, out).    % 46

pos(_ nci _, inp).   % 47
pos(_ nin _, inp).   % 48
pos(_ ndp _, out).   % 49

pos(-_, out).        % 50

pos(A, inp) :- primitive(A, _), atfoc(inp).
pos(A, out) :- primitive(A, _), atfoc(out).

%p2r([l(A, Phi, [])], f(A, Phi)), prf(id, [l(A, Phi, [])], f(A), [])) :- primitive(A, []).
%
%p2r([l(A, Phi, [[1]])], f(A, Phi), prf(id, [l(A, Phi, [[1]])], f(A), [])) :- primitive(A, [1]).

p2r([]: [l(A, Phi, [])], f(A, Phi), prf(id, []: [l(A, Phi, [])], f(A), [])) :-
      atfoc(out),
      primitive(A, []).

p2r([]: [l(A, Phi, [[1]])], f(A, Phi), prf(id, []: [l(A, Phi, [[1]])], f(A), [])) :-
    atfoc(out),
    primitive(A, [1]).

% 3

p2r(Stoup: Delta, f(P1*P2, [pair, Phi, Psi]), prf(right(product), Stoup: Delta, f(P1*P2), [SubPrf1, SubPrf2])) :-
```

```
      pos(P1, out), pos(P2, out),
      ssort(P1, SP1), (!),
      partition(Stoup, Stoup1, Stoup2),
      append(Gamma1, Gamma2, Delta),
      configsort(Gamma1, SP1),
      p2r(Stoup1: Gamma1, f(P1, Phi), SubPrf1),
      p2r(Stoup2: Gamma2, f(P2, Psi), SubPrf2).

p2r(Stoup: Delta, f(P*N, [pair, Phi, Psi]), prf(right(product), Stoup: Delta, f(P*N), [SubPrf1, SubPrf2])) :-
      pos(P, out), neg(N, out),
      ssort(P, SP), (!),
      partition(Stoup, Stoup1, Stoup2),
      append(Gamma1, Gamma2, Delta),
      configsort(Gamma1, SP),
      p2r(Stoup1: Gamma1, f(P, Phi), SubPrf1),
      p1(Stoup2: Gamma2, l(N, Psi), SubPrf2).

p2r(Stoup: Delta, f(N*P, [pair, Phi, Psi]), prf(right(product), Stoup: Delta, f(N*P), [SubPrf1, SubPrf2])) :-
      neg(N, out), pos(P, out),
      ssort(N, SN), (!),
      partition(Stoup, Stoup1, Stoup2),
      append(Gamma1, Gamma2, Delta),
      configsort(Gamma1, SN),
      p1(Stoup1: Gamma1, l(N, Phi), SubPrf1),
      p2r(Stoup2: Gamma2, f(P, Psi), SubPrf2).

p2r(Stoup: Delta, f(N1*N2, [pair, Phi, Psi]), prf(right(product), Stoup: Delta, f(N1*N2), [SubPrf1, SubPrf2])) :-
      neg(N1, out), neg(N2, out),
      ssort(N1, SN1), (!),
      partition(Stoup, Stoup1, Stoup2),
      append(Gamma1, Gamma2, Delta),
      configsort(Gamma1, SN1),
      p1(Stoup1: Gamma1, l(N1, Phi), SubPrf1),
      p1(Stoup2: Gamma2, l(N2, Psi), SubPrf2).

% 4

p2r([]: [], f(i, 0), prf(right(productunit), []: [], f(i), [])).

% 7+

p2r(Stoup: Delta, f(P1 dp P2, [pair, Phi, Psi]), prf(right(dprod), Stoup: Delta, f(P1 dp P2), [SubPrf1, SubPrf2])) :-
      pos(P1, out), pos(P2, out),
      configsort(Delta, S),
      do_ones(S, Ls), (!),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, [Gamma2|Ls], Delta),
      p2r(Stoup1: Gamma1, f(P1, Phi), SubPrf1),
      p2r(Stoup2: Gamma2, f(P2, Psi), SubPrf2).

p2r(Stoup: Delta, f(P dp N, [pair, Phi, Psi]), prf(right(dprod), Stoup: Delta, f(P dp N), [SubPrf1, SubPrf2])) :-
      pos(P, out), neg(N, out),
      configsort(Delta, S),
      do_ones(S, Ls), (!),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, [Gamma2|Ls], Delta),
      p2r(Stoup1: Gamma1, f(P, Phi), SubPrf1),
      p1(Stoup2: Gamma2, l(N, Psi), SubPrf2).

p2r(Stoup: Delta, f(N dp P, [pair, Phi, Psi]), prf(right(dprod), Stoup: Delta, f(N dp P), [SubPrf1, SubPrf2])) :-
      neg(N, out), pos(P, out),
      configsort(Delta, S),
      do_ones(S, Ls), (!),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, [Gamma2|Ls], Delta),
      p1(Stoup1: Gamma1, l(N, Phi), SubPrf1),
      p2r(Stoup2: Gamma2, f(P, Psi), SubPrf2).

p2r(Stoup: Delta, f(N1 dp N2, [pair, Phi, Psi]), prf(right(dprod), Stoup: Delta, f(N1 dp N2), [SubPrf1, SubPrf2])) :-
      neg(N1, out), neg(N2, out),
```

```
    configsort(Delta, S),
    do_ones(S, Ls), (!),
    partition(Stoup, Stoup1, Stoup2),
    fold([1|S], Gamma1, [Gamma2|Ls], Delta),
    p1(Stoup1: Gamma1, 1(N1, Phi)), SubPrf1),
    p1(Stoup2: Gamma2, 1(N2, Psi)), SubPrf2).

% 7-

p2r(Stoup: Delta, f(P1 dpn P2, [pair, Phi, Psi]), prf(right(dprodn), Stoup: Delta, f(P1 dpn P2), [SubPrf1, SubPrf2])) :-
      pos(P1, out), pos(P2, out),
      configsort(Delta, S),
      do_ones(S, Ls),
      append(Ls, [[Gamma2]], LsGamma2), (!),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, LsGamma2, Delta),
      p2r(Stoup1: Gamma1, f(P1, Phi)), SubPrf1),
      p2r(Stoup2: Gamma2, f(P2, Psi)), SubPrf2).

p2r(Stoup: Delta, f(P dpn N, [pair, Phi, Psi]), prf(right(dprodn), Stoup: Delta, f(P dpn N), [SubPrf1, SubPrf2])) :-
    pos(P, out), neg(N, out),
      configsort(Delta, S),
      do_ones(S, Ls),
      append(Ls, [[Gamma2]], LsGamma2), (!),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, LsGamma2, Delta),
      p2r(Stoup1: Gamma1, f(P, Phi)), SubPrf1),
      p1(Stoup2: Gamma2, 1(N, Psi)), SubPrf2).

p2r(Stoup: Delta, f(N dpn P, [pair, Phi, Psi]), prf(right(dprodn), Stoup: Delta, f(N dpn P), [SubPrf1, SubPrf2])) :-
      neg(N, out), pos(P, out),
      configsort(Delta, S),
      do_ones(S, Ls),
      append(Ls, [[Gamma2]], LsGamma2), (!),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, LsGamma2, Delta),
      p1(Stoup1: Gamma1, 1(N, Phi)), SubPrf1),
      p2r(Stoup2: Gamma2, f(P, Psi)), SubPrf2).

p2r(Stoup: Delta, f(N1 dpn N2, [pair, Phi, Psi]), prf(right(dprodn), Stoup: Delta, f(N1 dpn N2), [SubPrf1, SubPrf2])) :-
      neg(N1, out), neg(N2, out),
      configsort(Delta, S),
      do_ones(S, Ls),
      append(Ls, [[Gamma2]], LsGamma2), (!),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, LsGamma2, Delta),
      p1(Stoup1: Gamma1, 1(N1, Phi)), SubPrf1),
      p1(Stoup2: Gamma2, 1(N2, Psi)), SubPrf2).

% 8

p2r([]: [1], f(j, 0), prf(right(dproductunit), []: [1], f(j), [])).

% 10

p2r(OMEGA, f(P+B, [1, Phi]), prf(right(adisj), OMEGA, f(P+B), [SubPrf])) :-
      pos(P, out),
      p2r(OMEGA, f(P, Phi), SubPrf).

p2r(OMEGA, f(N+B, [1, Phi]), prf(right(adisj), OMEGA, f(N+B), [SubPrf])) :-
      neg(N, out),
      p1(OMEGA, 1(N, Phi), SubPrf).

p2r(OMEGA, f(A+P, [2, Psi]), prf(right(adisj), OMEGA, f(A+P), [SubPrf])) :-
    pos(P, out),
      p2r(OMEGA, f(P, Psi), SubPrf).

p2r(OMEGA, f(A+N, [2, Psi]), prf(right(adisj), OMEGA, f(A+N), [SubPrf])) :- (!),
      neg(N, out),
      p1(OMEGA, 1(N, Psi), SubPrf).
```

```
% 12

p2r(OMEGA, f(V e P, [pair, T, Phi]), prf(right(exstq), OMEGA, f(V e P), [SubPrf])) :-
      pos(P, out),
      tsubst(T, V, P, PP), (!),
      p2r(OMEGA, f(PP, Phi), SubPrf).

p2r(OMEGA, f(V e N, [pair, T, Phi]), prf(right(exstq), OMEGA, f(V e N), [SubPrf])) :-
      neg(N, out),
      tsubst(T, V, N, NN), (!),
      p1(OMEGA, l(NN, Phi), SubPrf).

% 14

p2r(OMEGA, f('M'P, [cup, Phi]), prf(right(emod), OMEGA, f('M'P), [SubPrf])) :-
      pos(P, out), (!),
      p2r(OMEGA, f(P, Phi), SubPrf).

p2r(OMEGA, f('M'N, [cap, Phi]), prf(right(emod), OMEGA, f('M'N), [SubPrf])) :- (!),
      neg(N, out),
      p1(OMEGA, l(N, Phi), SubPrf).

% 16

p2r([]:[b([]: Gamma)], f(br P, Phi), prf(right(brack), []:[b([]: Gamma)], f(br P), [SubPrf])) :-
      pos(P, out), (!),
      p2r([]: Gamma, f(P, Phi), SubPrf).

p2r([]:[b([]: Gamma)], f(br N, Phi), prf(right(brack), []:[b([]: Gamma)], f(br N), [SubPrf])) :- (!),
      neg(N, out),
      p1([]: Gamma, l(N, Phi), SubPrf).

% 18

p2r(OMEGA, f('?'P, [Phi]), prf(right(eexp), OMEGA, f('?'P), [SubPrf])) :-
      pos(P, out),
      p2r(OMEGA, f(P, Phi), SubPrf).

p2r(OMEGA, f('?'N, [Phi]), prf(right(eexp), OMEGA, f('?'N), [SubPrf])) :-
      neg(N, out),
      p1(OMEGA, l(N, Phi), SubPrf).

p2r(Stoup: Delta, f('?'P, [Phi|Psi]), prf(expan(eexp), Stoup: Delta, f('?'P), [SubPrf1, SubPrf2])) :-
      pos(P, out), (!),
      partition(Stoup, Stoup1, Stoup2),
      append(Delta1, Delta2, Delta),
      p2r(Stoup1: Delta1, f(P, Phi), SubPrf1),
      p2r(Stoup2: Delta2, f('?'P, Psi), SubPrf2).

p2r(Stoup: Delta, f('?'N, [Phi|Psi]), prf(expan(eexp), Stoup: Delta, f('?'N), [SubPrf1, SubPrf2])) :- (!),
      neg(N, out),
      partition(Stoup, Stoup1, Stoup2),
      append(Delta1, Delta2, Delta),
      p1(Stoup1: Delta1, l(N, Phi), SubPrf1),
      p1(Stoup2: Delta2, l('?'N, Psi), SubPrf2).

% 24

p2r(Stoup: Delta, f(P1 lip P2, Psi), prf(right(liproduct), Stoup: Delta, f(P1 lip P2), [SubPrf1, SubPrf2])) :-
      pos(P1, out), pos(P2, out),
      ssort(P1, SP1), (!),
      partition(Stoup, Stoup1, Stoup2),
      append(Gamma1, Gamma2, Delta),
      configsort(Gamma1, SP1),
      p2r(Stoup1: Gamma1, f(P1, _), SubPrf1),
      p2r(Stoup2: Gamma2, f(P2, Psi), SubPrf2).

p2r(Stoup: Delta, f(P lip N, Psi), prf(right(liproduct), Stoup: Delta, f(P lip N), [SubPrf1, SubPrf2])) :-
      pos(P, out), neg(N, out),
```

```
    ssort(P, SP), (!),
    partition(Stoup, Stoup1, Stoup2),
    append(Gamma1, Gamma2, Delta),
    configsort(Gamma1, SP),
    p2r(Stoup1: Gamma1, f(P, _), SubPrf1),
    p1(Stoup2: Gamma2, l(N, Psi), SubPrf2).

p2r(Stoup: Delta, f(N lip P, Psi), prf(right(liproduct), Stoup: Delta, f(N lip P), [SubPrf1, SubPrf2])) :-
    neg(N, out), pos(P, out),
    ssort(N, SN), (!),
    partition(Stoup, Stoup1, Stoup2),
    append(Gamma1, Gamma2, Delta),
    configsort(Gamma1, SN),
    p1(Stoup1: Gamma1, l(N, _), SubPrf1),
    p2r(Stoup2: Gamma2, f(P, Psi), SubPrf2).

p2r(Stoup: Delta, f(N1 lip N2, Psi), prf(right(liproduct), Stoup: Delta, f(N1 lip N2), [SubPrf1, SubPrf2])) :-
    neg(N1, out), neg(N2, out),
    ssort(N1, SN1), (!),
    partition(Stoup, Stoup1, Stoup2),
    append(Gamma1, Gamma2, Delta),
    configsort(Gamma1, SN1),
    p1(Stoup1: Gamma1, l(N1, _), SubPrf1),
    p1(Stoup2: Gamma2, l(N2, Psi), SubPrf2).

% 25

p2r(Stoup: Delta, f(P1 rip P2, Phi), prf(right(riproduct), Stoup: Delta, f(P1 rip P2), [SubPrf1, SubPrf2])) :-
    pos(P1, out), pos(P2, out),
    ssort(P1, SP1), (!),
    partition(Stoup, Stoup1, Stoup2),
    append(Gamma1, Gamma2, Delta),
    configsort(Gamma1, SP1),
    p2r(Stoup1: Gamma1, f(P1, Phi), SubPrf1),
    p2r(Stoup2: Gamma2, f(P2, _), SubPrf2).

p2r(Stoup: Delta, f(P rip N, Phi), prf(right(riproduct), Stoup: Delta, f(P rip N), [SubPrf1, SubPrf2])) :-
    pos(P, out), neg(N, out),
    ssort(P, SP), (!),
    partition(Stoup, Stoup1, Stoup2),
    append(Gamma1, Gamma2, Delta),
    configsort(Gamma1, SP),
    p2r(Stoup1: Gamma1, f(P, Phi), SubPrf1),
    p1(Stoup2: Gamma2, l(N, _), SubPrf2).

p2r(Stoup: Delta, f(N rip P, Phi), prf(right(riproduct), Stoup: Delta, f(N rip P), [SubPrf1, SubPrf2])) :-
    neg(N, out), pos(P, out),
    ssort(N, SN), (!),
    partition(Stoup, Stoup1, Stoup2),
    append(Gamma1, Gamma2, Delta),
    configsort(Gamma1, SN),
    p1(Stoup1: Gamma1, l(N, Phi), SubPrf1),
    p2r(Stoup2: Gamma2, f(P, _), SubPrf2).

p2r(Stoup: Delta, f(N1 rip N2, Phi), prf(right(riproduct), Stoup: Delta, f(N1 rip N2), [SubPrf1, SubPrf2])) :-
    neg(N1, out), neg(N2, out),
    ssort(N1, SN1), (!),
    partition(Stoup, Stoup1, Stoup2),
    append(Gamma1, Gamma2, Delta),
    configsort(Gamma1, SN1),
    p1(Stoup1: Gamma1, l(N1, Phi), SubPrf1),
    p1(Stoup2: Gamma2, l(N2, _), SubPrf2).

% 30+

p2r(Stoup: Delta, f(P1 uidp P2, Psi), prf(right(uidprod), Stoup: Delta, f(P1 uidp P2), [SubPrf1, SubPrf2])) :-
    pos(P1, out), pos(P2, out),
    configsort(Delta, S),
    do_ones(S, Ls), (!),
    partition(Stoup, Stoup1, Stoup2),
```

```
      fold([1|S], Gamma1, [Gamma2|Ls], Delta),
      p2r(Stoup1: Gamma1, f(P1, _), SubPrf1),
      p2r(Stoup2: Gamma2, f(P2, Psi), SubPrf2).

p2r(Stoup: Delta, f(P uidp N, Psi), prf(right(uidprod), Stoup: Delta, f(P uidp N), [SubPrf1, SubPrf2])) :-
      pos(P, out), neg(N, out),
      configsort(Delta, S),
      do_ones(S, Ls), (!),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, [Gamma2|Ls], Delta),
      p2r(Stoup1: Gamma1, f(P, _), SubPrf1),
      p1(Stoup2: Gamma2, l(N, Psi), SubPrf2).

p2r(Stoup: Delta, f(N uidp P, Psi), prf(right(uidprod), Stoup: Delta, f(N uidp P), [SubPrf1, SubPrf2])) :-
      neg(N, out), pos(P, out),
      configsort(Delta, S),
      do_ones(S, Ls), (!),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, [Gamma2|Ls], Delta),
      p1(Stoup1: Gamma1, l(N, _), SubPrf1),
      p2r(Stoup2: Gamma2, f(P, Psi), SubPrf2).

p2r(Stoup: Delta, f(N1 uidp N2, Psi), prf(right(uidprod), Stoup: Delta, f(N1 uidp N2), [SubPrf1, SubPrf2])) :-
      neg(N1, out), neg(N2, out),
      configsort(Delta, S),
      do_ones(S, Ls), (!),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, [Gamma2|Ls], Delta),
      p1(Stoup1: Gamma1, l(N1, _), SubPrf1),
      p1(Stoup2: Gamma2, l(N2, Psi), SubPrf2).

% 30-

p2r(Stoup: Delta, f(P1 uidpn P2, Psi), prf(right(uidprodn), Stoup: Delta, f(P1 uidpn P2), [SubPrf1, SubPrf2])) :-
      pos(P1, out), pos(P2, out),
      configsort(Delta, S),
      do_ones(S, Ls),
      append(Ls, [[Gamma2]], LsGamma2), (!),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, LsGamma2, Delta),
      p2r(Stoup1: Gamma1, f(P1, _), SubPrf1),
      p2r(Stoup2: Gamma2, f(P2, Psi), SubPrf2).

p2r(Stoup: Delta, f(P uidpn N, Psi), prf(right(uidprodn), Stoup: Delta, f(P uidpn N), [SubPrf1, SubPrf2])) :-
      pos(P, out), neg(N, out),
      configsort(Delta, S),
      do_ones(S, Ls),
      append(Ls, [[Gamma2]], LsGamma2), (!),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, LsGamma2, Delta),
      p2r(Stoup1: Gamma1, f(P, _), SubPrf1),
      p1(Stoup2: Gamma2, l(N, Psi), SubPrf2).

p2r(Stoup: Delta, f(N uidpn P, Psi), prf(right(uidprodn), Stoup: Delta, f(N uidpn P), [SubPrf1, SubPrf2])) :-
      neg(N, out), pos(P, out),
      configsort(Delta, S),
      do_ones(S, Ls),
      append(Ls, [[Gamma2]], LsGamma2), (!),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, LsGamma2, Delta),
      p1(Stoup1: Gamma1, l(N, _), SubPrf1),
      p2r(Stoup2: Gamma2, f(P, Psi), SubPrf2).

p2r(Stoup: Delta, f(N1 uidpn N2, Psi), prf(right(uidprodn), Stoup: Delta, f(N1 uidpn N2), [SubPrf1, SubPrf2])) :-
      neg(N1, out), neg(N2, out),
      configsort(Delta, S),
      do_ones(S, Ls),
      append(Ls, [[Gamma2]], LsGamma2), (!),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, LsGamma2, Delta),
```

```
    p1(Stoup1: Gamma1, 1(N1, _), SubPrf1),
    p1(Stoup2: Gamma2, 1(N2, Psi), SubPrf2).

% 31+

p2r(Stoup: Delta, f(P1 lidp P2, Phi), prf(right(lidprod), Stoup: Delta, f(P1 lidp P2), [SubPrf1, SubPrf2])) :-
    pos(P1, out), pos(P2, out),
    configsort(Delta, S),
    do_ones(S, Ls), (!),
    partition(Stoup, Stoup1, Stoup2),
    fold([1|S], Gamma1, [Gamma2|Ls], Delta),
    p2r(Stoup1: Gamma1, f(P1, Phi), SubPrf1),
    p2r(Stoup2: Gamma2, f(P2, _), SubPrf2).

p2r(Stoup: Delta, f(P lidp N, Phi), prf(right(lidprod), Stoup: Delta, f(P lidp N), [SubPrf1, SubPrf2])) :-
    pos(P, out), neg(N, out),
    configsort(Delta, S),
    do_ones(S, Ls), (!),
    partition(Stoup, Stoup1, Stoup2),
    fold([1|S], Gamma1, [Gamma2|Ls], Delta),
    p2r(Stoup1: Gamma1, f(P, Phi), SubPrf1),
    p1(Stoup2: Gamma2, 1(N, _), SubPrf2).

p2r(Stoup: Delta, f(N lidp P, Phi), prf(right(lidprod), Stoup: Delta, f(N lidp P), [SubPrf1, SubPrf2])) :-
    neg(N, out), pos(P, out),
    configsort(Delta, S),
    do_ones(S, Ls), (!),
    partition(Stoup, Stoup1, Stoup2),
    fold([1|S], Gamma1, [Gamma2|Ls], Delta),
    p1(Stoup1: Gamma1, 1(N, Phi), SubPrf1),
    p2r(Stoup2: Gamma2, f(P, _), SubPrf2).

p2r(Stoup: Delta, f(N1 lidp N2, Phi), prf(right(lidprod), Stoup: Delta, f(N1 lidp N2), [SubPrf1, SubPrf2])) :-
    neg(N1, out), neg(N2, out),
    configsort(Delta, S),
    do_ones(S, Ls), (!),
    partition(Stoup, Stoup1, Stoup2),
    fold([1|S], Gamma1, [Gamma2|Ls], Delta),
    p1(Stoup1: Gamma1, 1(N1, Phi), SubPrf1),
    p1(Stoup2: Gamma2, 1(N2, _), SubPrf2).

% 31-

p2r(Stoup: Delta, f(P1 lidpn P2, Phi), prf(right(lidprodn), Stoup: Delta, f(P1 lidpn P2), [SubPrf1, SubPrf2])) :-
    pos(P1, out), pos(P2, out),
    configsort(Delta, S),
    do_ones(S, Ls),
    append(Ls, [[Gamma2]], LsGamma2), (!),
    partition(Stoup, Stoup1, Stoup2),
    fold([1|S], Gamma1, LsGamma2, Delta),
    p2r(Stoup1: Gamma1, f(P1, Phi), SubPrf1),
    p2r(Stoup2: Gamma2, f(P2, _), SubPrf2).

p2r(Stoup: Delta, f(P lidpn N, Phi), prf(right(lidprodn), Stoup: Delta, f(P lidpn N), [SubPrf1, SubPrf2])) :-
    pos(P, out), neg(N, out),
    configsort(Delta, S),
    do_ones(S, Ls),
    append(Ls, [[Gamma2]], LsGamma2), (!),
    partition(Stoup, Stoup1, Stoup2),
    fold([1|S], Gamma1, LsGamma2, Delta),
    p2r(Stoup1: Gamma1, f(P, Phi), SubPrf1),
    p1(Stoup2: Gamma2, 1(N, _), SubPrf2).

p2r(Stoup: Delta, f(N lidpn P, Phi), prf(right(lidprodn), Stoup: Delta, f(N lidpn P), [SubPrf1, SubPrf2])) :-
    neg(N, out), pos(P, out),
    configsort(Delta, S),
    do_ones(S, Ls),
    append(Ls, [[Gamma2]], LsGamma2), (!),
    partition(Stoup, Stoup1, Stoup2),
    fold([1|S], Gamma1, LsGamma2, Delta),
```

```
      p1(Stoup1: Gamma1, l(N, Phi), SubPrf1),
      p2r(Stoup2: Gamma2, f(P, _), SubPrf2).

p2r(Stoup: Delta, f(N1 lidpn N2, Phi), prf(right(lidprodn), Stoup: Delta, f(N1 lidpn N2), [SubPrf1, SubPrf2])) :-
      neg(N1, out), neg(N2, out),
      configsort(Delta, S),
      do_ones(S, Ls),
      append(Ls, [[Gamma2]], LsGamma2), (!),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, LsGamma2, Delta),
      p1(Stoup1: Gamma1, l(N1, Phi), SubPrf1),
      p1(Stoup2: Gamma2, l(N2, _), SubPrf2).

% 32

p2r([]: [], f(w([]), 0), prf(right(words), []: [], f(w([])), [])).

% 34

p2r(OMEGA, f(P iad B, Phi), prf(right(iadisj), OMEGA, f(P iad B), [SubPrf])) :-
      pos(P, out),
      p2r(OMEGA, f(P, Phi), SubPrf).

p2r(OMEGA, f(N iad B, Phi), prf(right(iadisj), OMEGA, f(N iad B), [SubPrf])) :-
      neg(N, out),
      p1(OMEGA, l(N, Phi), SubPrf).

p2r(OMEGA, f(A iad P, Psi), prf(right(iadisj), OMEGA, f(A iad P), [SubPrf])) :-
      pos(P, out),
      p2r(OMEGA, f(P, Psi), SubPrf).

p2r(OMEGA, f(A iad N, Psi), prf(right(iadisj), OMEGA, f(A iad N), [SubPrf])) :-
      neg(N, out), (!),
      p1(OMEGA, l(N, Psi), SubPrf).

% 36

p2r(OMEGA, f(V ie P, Phi), prf(right(iexstq), OMEGA, f(V ie P), [SubPrf])) :-
      pos(P, out),
      tsubst(_, V, P, PP), (!),
      p2r(OMEGA, f(PP, Phi), SubPrf).

p2r(OMEGA, f(V ie N, Phi), prf(right(iexstq), OMEGA, f(V ie N), [SubPrf])) :-
      neg(N, out),
      tsubst(_, V, N, NN), (!),
      p1(OMEGA, l(NN, Phi), SubPrf).

% 38

p2r(OMEGA, f(iM P, Phi), prf(right(iemod), OMEGA, f(iM P), [SubPrf])) :-
      pos(P, out), (!),
      p2r(OMEGA, f(P, Phi), SubPrf).

p2r(OMEGA, f(iM N, Phi), prf(right(iemod), OMEGA, f(iM N), [SubPrf])) :- (!),
      neg(N, out),
      p1(OMEGA, l(N, Phi), SubPrf).

% 41

p2r(Stoup: Gamma1, f(li P, Phi), prf(right(linj), Stoup: Gamma1, f(li P), [SubPrf])) :-
      pos(P, out),
      append(Gamma, [1], Gamma1), (!),
      p2r(Stoup: Gamma, f(P, Phi), SubPrf).

p2r(Stoup: Gamma1, f(li N, Phi), prf(right(linj), Stoup: Gamma1, f(li N), [SubPrf])) :-
%      neg(N, out),
      append(Gamma, [1], Gamma1), (!),
      p2r(Stoup: Gamma, f(N, Phi), SubPrf).

% 42
```

```
p2r(Stoup: [1|Gamma], f(ri P, Phi), prf(right(rinj), Stoup: [1|Gamma], f(ri P), [SubPrf])) :-
        pos(P, out), (!),
        p2r(Stoup: Gamma, f(P, Phi), SubPrf).

p2r(Stoup: [1|Gamma], f(ri N, Phi), prf(right(rinj), Stoup: [1|Gamma], f(ri N), [SubPrf])) :- (!),
    neg(N, out),
        p2r(Stoup: Gamma, f(N, Phi), SubPrf).

% 44+

p2r(Stoup: Delta, f(bg P, Phi), prf(right(bridge), Stoup: Delta, f(bg P), [SubPrf])) :-
        pos(P, out),
        configsort(Delta, S),
        do_ones(S, Ls), (!),
        fold([1|S], Gamma1, [[]|Ls], Delta),
        p2r(Stoup: Gamma1, f(P, Phi), SubPrf).

p2r(Stoup: Delta, f(bg N, Phi), prf(right(bridge), Stoup: Delta, f(bg N), [SubPrf])) :-
        neg(N, out),
        configsort(Delta, S),
        do_ones(S, Ls), (!),
        fold([1|S], Gamma1, [[]|Ls], Delta),
        p1(Stoup: Gamma1, 1(N, Phi), SubPrf).

% 44-

p2r(Stoup: Delta, f(bgn P, Phi), prf(right(bridgen), Stoup: Delta, f(bgn P), [SubPrf])) :-
        pos(P, out),
        configsort(Delta, S),
        do_ones(S, Ls),
        append(Ls, [[]], Lss), (!),
        fold([1|S], Gamma1, Lss, Delta),
        p2r(Stoup: Gamma1, f(P, Phi), SubPrf).

p2r(Stoup: Delta, f(bgn N, Phi), prf(right(bridgen), Stoup: Delta, f(bgn N), [SubPrf])) :-
    neg(N, out),
        configsort(Delta, S),
        do_ones(S, Ls);
        append(Ls, [[]], Lss), (!),
        fold([1|S], Gamma1, Lss, Delta),
        p1(Stoup: Gamma1, 1(N, Phi), SubPrf).

% 46

p2r(Stoup: Delta, f(P1 np P2, [pair, Phi, Psi]), prf(right(nprod), Stoup: Delta, f(P1 np P2), [SubPrf1, SubPrf2])) :-
        pos(P1, out), pos(P2, out),
        ssort(P1, SP1),
        partition(Stoup, Stoup1, Stoup2),
        append(Gamma1, Gamma2, Delta),
        configsort(Gamma1, SP1),
        p2r(Stoup1: Gamma1, f(P1, Phi), SubPrf1),
        p2r(Stoup2: Gamma2, f(P2, Psi), SubPrf2).

p2r(Stoup: Delta, f(P np N, [pair, Phi, Psi]), prf(right(nprod), Stoup: Delta, f(P np N), [SubPrf1, SubPrf2])) :-
        pos(P, out), neg(N, out),
        ssort(P, SP),
        partition(Stoup, Stoup1, Stoup2),
        append(Gamma1, Gamma2, Delta),
        configsort(Gamma1, SP),
        p2r(Stoup1: Gamma1, f(P, Phi), SubPrf1),
        p1(Stoup2: Gamma2, 1(N, Psi), SubPrf2).

p2r(Stoup: Delta, f(N np P, [pair, Phi, Psi]), prf(right(nprod), Stoup: Delta, f(N np P), [SubPrf1, SubPrf2])) :-
        neg(N, out), pos(P, out),
        ssort(N, SN),
        partition(Stoup, Stoup1, Stoup2),
        append(Gamma1, Gamma2, Delta),
        configsort(Gamma1, SN),
        p1(Stoup1: Gamma1, 1(N, Phi), SubPrf1),
```

```
      p2r(Stoup2: Gamma2, f(P, Psi), SubPrf2).

p2r(Stoup: Delta, f(N1 np N2, [pair, Phi, Psi]), prf(right(nprod), Stoup: Delta, f(N1 np N2), [SubPrf1, SubPrf2])) :-
      neg(N1, out), neg(N2, out),
      ssort(N1, SN1),
      partition(Stoup, Stoup1, Stoup2),
      append(Gamma1, Gamma2, Delta),
      configsort(Gamma1, SN1),
      p1(Stoup1: Gamma1, l(N1, Phi), SubPrf1),
      p1(Stoup2: Gamma2, l(N2, Psi), SubPrf2).

p2r(Stoup: Delta, f(P1 np P2, [pair, Phi, Psi]), prf(right(nprod), Stoup: Delta, f(P1 np P2), [SubPrf1, SubPrf2])) :-
      pos(P1, out), pos(P2, out),
      ssort(P1, SP1),
      partition(Stoup, Stoup1, Stoup2),
      append(Gamma1, Gamma2, Delta),
      configsort(Gamma1, SP1), (!),
      p2r(Stoup1: Gamma1, f(P2, Psi), SubPrf1),
      p2r(Stoup2: Gamma2, f(P1, Phi), SubPrf2).

p2r(Stoup: Delta, f(P np N, [pair, Phi, Psi]), prf(right(nprod), Stoup: Delta, f(P np N), [SubPrf1, SubPrf2])) :-
      pos(P, out), neg(N, out),
      ssort(P, SP),
      partition(Stoup, Stoup1, Stoup2),
      append(Gamma1, Gamma2, Delta),
      configsort(Gamma1, SP), (!),
      p2r(Stoup1: Gamma1, f(N, Psi), SubPrf1),
      p1(Stoup2: Gamma2, l(P, Phi), SubPrf2).

p2r(Stoup: Delta, f(N np P, [pair, Phi, Psi]), prf(right(nprod), Stoup: Delta, f(N np P), [SubPrf1, SubPrf2])) :-
      neg(N, out), pos(P, out),
      ssort(N, SN),
      partition(Stoup, Stoup1, Stoup2),
      append(Gamma1, Gamma2, Delta),
      configsort(Gamma1, SN), (!),
      p1(Stoup1: Gamma1, l(P, Psi), SubPrf1),
      p2r(Stoup2: Gamma2, f(N, Phi), SubPrf2).

p2r(Stoup: Delta, f(N1 np N2, [pair, Phi, Psi]), prf(right(nprod), Stoup: Delta, f(N1 np N2), [SubPrf1, SubPrf2])) :-
      neg(N1, out), neg(N2, out),
      ssort(N1, SN1),
      partition(Stoup, Stoup1, Stoup2),
      append(Gamma1, Gamma2, Delta),
      configsort(Gamma1, SN1), (!),
      p1(Stoup1: Gamma1, l(N2, Psi), SubPrf1),
      p1(Stoup2: Gamma2, l(N1, Phi), SubPrf2).

% 49

p2r(Stoup: Delta, f(P1 ndp P2, [pair, Phi, Psi]), prf(right(ndprod), Stoup: Delta, f(P1 ndp P2), [SubPrf1, SubPrf2])) :-
      pos(P1, out), pos(P2, out),
      configsort(Delta, S),
      do_ones(S, Ls),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, [Gamma2|Ls], Delta),
      p2r(Stoup1: Gamma1, f(P1, Phi), SubPrf1),
      p2r(Stoup2: Gamma2, f(P2, Psi), SubPrf2).

p2r(Stoup: Delta, f(P ndp N, [pair, Phi, Psi]), prf(right(ndprod), Stoup: Delta, f(P ndp N), [SubPrf1, SubPrf2])) :-
      pos(P, out), neg(N, out),
      configsort(Delta, S),
      do_ones(S, Ls),
      partition(Stoup, Stoup1, Stoup2),
      fold([1|S], Gamma1, [Gamma2|Ls], Delta),
      p2r(Stoup1: Gamma1, f(P, Phi), SubPrf1),
      p1(Stoup2: Gamma2, l(N, Psi), SubPrf2).

p2r(Stoup: Delta, f(N ndp P, [pair, Phi, Psi]), prf(right(ndprod), Stoup: Delta, f(N ndp P), [SubPrf1, SubPrf2])) :-
      neg(N, out), pos(P, out),
      configsort(Delta, S),
```

```
    do_ones(S, Ls),
    partition(Stoup, Stoup1, Stoup2),
    fold([1|S], Gamma1, [Gamma2|Ls], Delta),
    p1(Stoup1: Gamma1, 1(N, Phi), SubPrf1),
    p2r(Stoup2: Gamma2, f(P, Psi), SubPrf2).

p2r(Stoup: Delta, f(N1 ndp N2, [pair, Phi, Psi]), prf(right(ndprod), Stoup: Delta, f(N1 ndp N2), [SubPrf1, SubPrf2])) :-
    neg(N1, out), neg(N2, out),
    configsort(Delta, S),
    do_ones(S, Ls), (!),
    partition(Stoup, Stoup1, Stoup2),
    fold([1|S], Gamma1, [Gamma2|Ls], Delta),
    p1(Stoup1: Gamma1, 1(N1, Phi), SubPrf1),
    p1(Stoup2: Gamma2, 1(N2, Psi), SubPrf2).

p2r(Stoup: Delta, f(P1 ndp P2, [pair, Phi, Psi]), prf(right(ndprod), Stoup: Delta, f(P1 ndp P2), [SubPrf1, SubPrf2])) :-
    pos(P1, out), pos(P2, out),
    configsort(Delta, S),
    do_ones(S, Ls),
    append(Ls, [[Gamma2]], LsGamma2), (!),
    partition(Stoup, Stoup1, Stoup2),
    fold([1|S], Gamma1, LsGamma2, Delta),
    p2r(Stoup1: Gamma1, f(P1, Phi), SubPrf1),
    p2r(Stoup2: Gamma2, f(P2, Psi), SubPrf2).

p2r(Stoup: Delta, f(P ndp N, [pair, Phi, Psi]), prf(right(ndprodn), Stoup: Delta, f(P ndp N), [SubPrf1, SubPrf2])) :-
    pos(P, out), neg(N, out),
    configsort(Delta, S),
    do_ones(S, Ls),
    append(Ls, [[Gamma2]], LsGamma2), (!),
    partition(Stoup, Stoup1, Stoup2);
    fold([1|S], Gamma1, LsGamma2, Delta),
    p2r(Stoup1: Gamma1, f(P, Phi)), SubPrf1),
    p1(Stoup2: Gamma2, 1(N, Psi), SubPrf2).

p2r(Stoup: Delta, f(N ndp P, [pair, Phi, Psi]), prf(right(ndprod), Stoup: Delta, f(N ndp P), [SubPrf1, SubPrf2])) :-
    neg(N, out), pos(P, out),
    configsort(Delta, S),
    do_ones(S, Ls),
    append(Ls, [[Gamma2]], LsGamma2), (!),
    partition(Stoup, Stoup1, Stoup2),
    fold([1|S], Gamma1, LsGamma2, Delta),
    p1(Stoup1: Gamma1, 1(N, Phi), SubPrf1),
    p2r(Stoup2: Gamma2, f(P, Psi), SubPrf2).

p2r(Stoup: Delta, f(N1 ndp N2, [pair, Phi, Psi]), prf(right(ndprod), Stoup: Delta, f(N1 ndp N2), [SubPrf1, SubPrf2])) :-
    neg(N1, out), neg(N2, out),
    configsort(Delta, S),
    do_ones(S, Ls),
    append(Ls, [[Gamma2]], LsGamma2),
    partition(Stoup, Stoup1, Stoup2),
    fold([1|S], Gamma1, LsGamma2, Delta),
    p1(Stoup1: Gamma1, 1(N1, Phi), SubPrf1),
    p1(Stoup2: Gamma2, 1(N2, Psi), SubPrf2).

% 50

p2r(OMEGA, f(P1-P2, Psi), prf(right(diff), OMEGA, f(P1-P2), [SubPrf])) :-
    pos(P1, out), pos(P2, out),
    p2r(OMEGA, f(P1, Psi), SubPrf),
    \+(p2r(OMEGA, f(P2, _), _)).

p2r(OMEGA, f(P-N, Psi), prf(right(diff), OMEGA, f(P-N), [SubPrf])) :-
    pos(P, out), neg(N, out),
    p2r(OMEGA, f(P, Psi), SubPrf),
    \+(p1(OMEGA, 1(N, _), _)).

p2r(OMEGA, f(N-P, Psi), prf(right(diff), OMEGA, 1(N-P), [SubPrf])) :-
    neg(N, out), pos(P, out),
    p1(OMEGA, 1(N, Psi), SubPrf),
```

```
      \+(p2r(OMEGA, f(P, _), _)).

p2r(OMEGA, f(N1-N2, Psi), prf(right(diff), OMEGA, l(N1-N2), [SubPrf])) :-
      neg(N1, out), neg(N2, out),
      p1(OMEGA, l(N1, Psi), SubPrf),
      \+(p1(OMEGA, l(N2, _), _)).

abindoff([], _, _, [], _).

abindoff([l(B lca A, Chi, Int)|Gamma], A, X, [l(B, [app, Chi, X], Int1)|Gamma1], F) :-
      abindofflist(Int, A, X, Int1, F),
      abindoff(Gamma, A, X, Gamma1, F).

abindoff([f(B lca A, Chi, Int)|Gamma], A, X, [l(B, [app, Chi, X], Int1)|Gamma1], defoc) :-
      abindofflist(Int, A, X, Int1, defoc),
      abindoff(Gamma, A, X, Gamma1, defoc).

abindoff([f(B lca A, Chi, Int)|Gamma], A, X, [f(B, [app, Chi, X], Int1)|Gamma1], foc) :-
      abindofflist(Int, A, X, Int1, foc),
      abindoff(Gamma, A, X, Gamma1, foc).

abindoff([l(A, Phi, Int)|Gamma], A, X, [l(A, Phi, Int)|Gamma1], F) :-
      abindoff(Gamma, A, X, Gamma1, F).

%abindoff([f(A, Phi, Int)|Gamma], A, X, [l(A, Phi, Int)|Gamma1], S, S2) :-
%      abindoff(Gamma, A, X, Gamma1, S, S2).

abindoff([b(Stoup: Gamma)|Delta], A, X, [b(Stoup: Gamma1)|Delta1], F) :-
      abindoff(Gamma, A, X, Gamma1, F),
      abindoff(Delta, A, X, Delta1, F).

abindofflist([], _, _, [], _).

abindofflist([H|T], A, X, [H1|T1], F) :-
      abindoff(H, A, X, H1, F),
      abindofflist(T, A, X, T1, F).

% tsubst(+Phi, +X, +Tin, -Tout) means that the result of substituting
% Phi for X in term/type Tin is Tout.

tsubst(_, _, V, V) :- var(V), (!).

tsubst(Phi, X, X, Phi) :- (!).

tsubst(_, _, C, C) :- (atom(C); integer(C)), (!).

tsubst(Phi, X, T1, T2) :-
      T1 =.. [H|T],
      tsubstlst(Phi, X, T, TT),
      T2 =.. [H|TT].

tsubstlst(_, _, [], []) :- (!).

tsubstlst(Phi, X, [H|T], [HH|TT]) :-
      tsubst(Phi, X, H, HH),
      tsubstlst(Phi, X, T, TT).

/*

tsubst(_, _, i, i).

tsubst(Phi, X, C/B, C1/B1) :- (!),
      tsubst(Phi, X, C, C1),
      tsubst(Phi, X, B, B1).

tsubst(Phi, X, A*B, A1*B1) :- (!),
      tsubst(Phi, X, A, A1),
      tsubst(Phi, X, B, B1).

tsubst(Phi, X, A bs C, A1 bs C1) :- (!),
```

```
    tsubst(Phi, X, A, A1),
    tsubst(Phi, X, C, C1).

tsubst(_, _, j, j) :- (!).

tsubst(Phi, X, C ci B, C1 ci B1) :- (!),
    tsubst(Phi, X, C, C1),
    tsubst(Phi, X, B, B1).

tsubst(Phi, X, A dp B, A1 dp B1) :- (!),
    tsubst(Phi, X, A, A1),
    tsubst(Phi, X, B, B1).

tsubst(Phi, X, A in C, A1 in C1) :- (!),
    tsubst(Phi, X, A, A1),
    tsubst(Phi, X, C, C1).

tsubst(Phi, X, C cin B, C1 cin B1) :- (!),
    tsubst(Phi, X, C, C1),
    tsubst(Phi, X, B, B1).

tsubst(Phi, X, A dpn B, A1 dpn B1) :- (!),
    tsubst(Phi, X, A, A1),
    tsubst(Phi, X, B, B1).

tsubst(Phi, X, A inn C, A1 inn C1) :- (!),
    tsubst(Phi, X, A, A1),
    tsubst(Phi, X, C, C1).

tsubst(Phi, X, 'L'A, 'L'A1) :- (!),
    tsubst(Phi, X, A, A1).

tsubst(Phi, X, iL A, iL A1) :- (!),
    tsubst(Phi, X, A, A1).

tsubst(Phi, X, br A, br A1) :- (!),
    tsubst(Phi, X, A, A1).

tsubst(Phi, X, ab A, ab A1) :- (!),
    tsubst(Phi, X, A, A1).

tsubst(Phi, X, A & B, A1 & B1) :- (!),
    tsubst(Phi, X, A, A1),
    tsubst(Phi, X, B, B1).

tsubst(Phi, X, A + B, A1 + B1) :- (!),
    tsubst(Phi, X, A, A1),
    tsubst(Phi, X, B, B1).

tsubst(Phi, X, A iac B, A1 iac B1) :- (!),
    tsubst(Phi, X, A, A1),
    tsubst(Phi, X, B, B1).

tsubst(Phi, X, A iad B, A1 iad B1) :- (!),
    tsubst(Phi, X, A, A1),
    tsubst(Phi, X, B, B1).

tsubst(_, X, X u A, X u A) :- (!).

tsubst(_, X, X e A, X e A) :- (!).

tsubst(_, X, X iu A, X iu A) :- (!).

tsubst(_, X, X ie A, X ie A) :- (!).

tsubst(Phi, X, Y u A, Y u A1) :-
    tsubst(Phi, X, A, A1).

tsubst(Phi, X, Y e A, Y e A1) :-
    tsubst(Phi, X, A, A1).
```

```
tsubst(Phi, X, Y iu A, Y iu A1) :-
      tsubst(Phi, X, A, A1).

tsubst(Phi, X, Y ie A, Y ie A1) :-
      tsubst(Phi, X, A, A1).

tsubst(Phi, X, B nd A, B1 nd A1) :- (!),
      tsubst(Phi, X, B, B1),
      tsubst(Phi, X, A, A1).

tsubst(Phi, X, C nci B, C1 nci B1) :- (!),
      tsubst(Phi, X, C, C1),
      tsubst(Phi, X, B, B1).

tsubst(Phi, X, A ndp B, A1 ndp B1) :- (!),
      tsubst(Phi, X, A, A1),
      tsubst(Phi, X, B, B1).

tsubst(Phi, X, A nin C, A1 nin C1) :- (!),
      tsubst(Phi, X, A, A1),
      tsubst(Phi, X, C, C1).

tsubst(Phi, X, li A, li A1) :- (!),
      tsubst(Phi, X, A, A1).

tsubst(Phi, X, lp A, lp A1) :- (!),
      tsubst(Phi, X, A, A1).

tsubst(Phi, X, bg A, bg A1) :- (!),
      tsubst(Phi, X, A, A1).

tsubst(Phi, X, sp A, sp A1) :- (!),
      tsubst(Phi, X, A, A1).

tsubst(Phi, X, bgn A, bgn A1) :- (!),
      tsubst(Phi, X, A, A1).

tsubst(Phi, X, spn A, spn A1) :- (!),
      tsubst(Phi, X, A, A1).

tsubst(Phi, X, rp A, rp A1) :- (!),
      tsubst(Phi, X, A, A1).

tsubst(Phi, X, ri A, ri A1) :- (!),
      tsubst(Phi, X, A, A1).

tsubst(Phi, X, '!'A, '!'A1) :- (!),
      tsubst(Phi, X, A, A1).

tsubst(Phi, X, '?' A, '?' A1) :- (!),
      tsubst(Phi, X, A, A1).

tsubst(Phi, X, B lca A, B1 lca A1) :- (!),
      tsubst(Phi, X, B, B1),
      tsubst(Phi, X, A, A1).

tsubst(Phi, X, -A, -A1) :- (!),
      tsubst(Phi, X, A, A1).

tsubst(Phi, X, A-B, A1-B1) :- (!),
      tsubst(Phi, X, A, A1),
      tsubst(Phi, X, B, B1).

% tsubst(_, X, forall2(X, L, A), forall2(X, L, A)) :- (!).

% tsubst(Phi, X, forall2(Y, L, A), forall2(Y, L2, A2)) :- (!),
%       tsubstlst(Phi, X, L, L2),
%       tsubst(Phi, X, A, A2).
```

```
tsubst(Phi, X, T, T1) :-
        T =.. [F|Args],
        tsubstlst(Phi, X, Args, Args1),
        T1 =.. [F|Args1].

tsubstlst(_, _, [], []).

tsubstlst(Phi, X, [H|T], [H1|T1]) :-
        tsubst(Phi, X, H, H1),
        tsubstlst(Phi, X, T, T1).

*/

% genins is like genwrap but only inserts individual focused items

genins([], Is, Is, []).

genins([1|Gamma], [1|Is], Isout, [1|Gamma1]) :-
        genins(Gamma, Is, Isout, Gamma1).

genins([1|Gamma], [f(A, Phi, Int)|Is], Isout, [f(A, Phi, Int)|Gamma1]) :-
        genins(Gamma, Is, Isout, Gamma1).

genins([1(A, Phi, Ls)|Gamma], Isin, Isout, [1(A, Phi, Ls1)|Gamma1]):-
        geninslst(Ls, Isin, Is, Ls1),
        genins(Gamma, Is, Isout, Gamma1).

genins([b(Stoup: Gamma)|Gamma1], Isin, Isout, [b(Stoup: Gamma2)|Gamma3]) :-
        genins(Gamma, Isin, Is, Gamma2),
        genins(Gamma1, Is, Isout, Gamma3).

geninslst([], Is, Is, []).

geninslst([L|Ls], Isin, Isout, [L1|Ls1]):-
        genins(L, Isin, Is, L1),
        geninslst(Ls, Is, Isout, Ls1).

% kappend(+S, -L1, -L2, +L) means that L is the concatenation of L1
% and L2 and the length of L1 is the same as that of the sort S.

kappend([], [], L, L).

kappend([1|S], [H|L1], L2, [H|L]) :-
        kappend(S, L1, L2, L).

configsort(Gamma, M) :-
        configsort(Gamma, M, []).

configsort([], M, M).

configsort([1|Gamma], [1|M1], M2) :-
        configsort(Gamma, M1, M2).

configsort([1(_,_,Ls)|Gamma], M1, M3) :-
        configsortlst(Ls, M1, M2),
        configsort(Gamma, M2, M3).

configsort([f(_,_,Ls)|Gamma], M1, M3) :-
        configsortlst(Ls, M1, M2),
        configsort(Gamma, M2, M3).

configsort([b(_: Gamma)|Gamma1], M1, M3) :-
        configsort(Gamma, M1, M2),
        configsort(Gamma1, M2, M3).

configsortlst([], M, M).

configsortlst([H|T], M1, M3) :-
        configsort(H, M1, M2),
        configsortlst(T, M2, M3).
```

```
% fold(+S, -Gamma, -Deltas, +Gamma1) means that configuration
% Gamma is of sort S and Gamma1 is the result of replacing
% in order the separators of Gamma by the configurations in Deltas.

fold(S, Gamma, Deltas, Gamma1) :-
      gfold(S, [], Gamma, Deltas, [], Gamma1).

% gfold(+Sin, ?Sout, -Gamma, -Deltasin, ?Deltasout, +Gamma1)
% means that configuration
% Gamma is of sort Sin-Sout and Gamma1 is the result of replacing
% in order the separators of Gamma by the configurations in Deltasin
% and Deltasout remain.

gfold(S, S, [], Deltas, Deltas, []).

gfold([1|S], Sout, [1|Gamma], [Delta|Deltas], Deltasout, Gamma1) :-
      append(Delta, Gamma2, Gamma1),
      gfold(S, Sout, Gamma, Deltas, Deltasout, Gamma2).

gfold(Sin, Sout, [l(A, Phi, Ls)|Gamma], Deltasin, Deltasout, [l(A, Phi, Ls1)|Gamma1]):-
      gfoldlst(Sin, S, Ls, Deltasin, Deltas, Ls1),
      gfold(S, Sout, Gamma, Deltas, Deltasout, Gamma1).

gfold(Sin, Sout, [f(A, Phi, Ls)|Gamma], Deltasin, Deltasout, [f(A, Phi, Ls1)|Gamma1]):-
      gfoldlst(Sin, S, Ls, Deltasin, Deltas, Ls1),
      gfold(S, Sout, Gamma, Deltas, Deltasout, Gamma1).

gfold(Sin, Sout, [b(Stoup: Gamma)|Gamma1], Deltasin, Deltasout, [b(Stoup: Gamma2)|Gamma3]) :-
      gfold(Sin, S, Gamma, Deltasin, Deltas, Gamma2),
      gfold(S, Sout, Gamma1, Deltas, Deltasout, Gamma3).

gfoldlst(S, S, [], Deltas, Deltas, []).

gfoldlst(Sin, Sout, [L|Ls], Deltasin, Deltasout, [L1|Ls1]):-
      gfold(Sin, S, L, Deltasin, Deltas, L1),
      gfoldlst(S, Sout, Ls, Deltas, Deltasout, Ls1).

% lexical insertion

lookup([], []).

lookup(Ws, [X|Xs]) :-
      append(W1, W2, Ws),
      lookup1(W1, X),
      lookup(W2, Xs).

lookup1(Ws, l(A, Phi, [])) :-
      lex(Ws, A, Phi).

lookup1(Ws, l(A, Phi, [Gamma])) :-
      nappend([W1, W2, W3], Ws),
      append(W1, [1|W3], W4),
      lex(W4, A, Phi),
      lookup(W2, Gamma).

lookup1(Ws, l(A, Phi, [[1]])) :-
      append(W1, W3, Ws),
      append(W1, [1|W3], W4),
      lex(W4, A, Phi).

lookup1([b(Ws)], b([]: Gamma)) :-
      lookup(Ws, Gamma).

% ppsem(+out, +Phi) pretty prints semantic form Phi.

% ppsem(user, [up, Phi]) :- (!), write(' '),
%       ppsem(user, Phi).

% ppsem(latex(S), [up, Phi]) :- (!), write(S, `\mbox{\^{ }}`),
```

```
%     ppsem(latex(S), Phi).

% ppsem(user, [dn, Phi]) :- (!), write('v'),
%       ppsem(user, Phi).

% ppsem(latex(S), [dn, Phi]) :- (!), write(S, '\mbox{\vf }}'),
%      ppsem(latex(S), Phi).

ppsem(user, [app, Phi, Psi]) :- (!), write('('),
ppsem(user, Phi),
write(' '),
ppsem(user, Psi),
write(')').

ppsem(latex(S), [app, Phi, Psi]) :- (!), write(S, '('),
ppsem(latex(S), Phi),
write(S, '\ '),
ppsem(latex(S), Psi),
write(S, ')').

ppsem(user, [pair, Phi, Psi]) :- (!), write('('),
ppsem(user, Phi),
write(', '),
ppsem(user, Psi),
write(')').

ppsem(user, [case, Chi, X, Phi, Y, Psi]) :- (!),
      write('('),
          ppsem(user, Chi),
          write('->'),
          write(X),
          write('.'),
          ppsem(user, Phi),
          write(';'),
          write(Y),
          write('.'),
          ppsem(user, Psi),
          write(')').

ppsem(latex(S), [case, Chi, X, Phi, Y, Psi]) :- (!),
          write(S, '('),
          ppsem(latex(S), Chi),
          write(S, '\rightarrow '),
          write(S, X),
          write(S, '.'),
          ppsem(latex(S), Phi),
          write(S, ' ; '),
          write(S, Y),
          write(S, '.'),
          ppsem(latex(S), Psi),
          write(S, ')').

ppsem(latex(S), [pair, Phi, Psi]) :- (!), write(S, '('),
 ppsem(latex(S), Phi),
 write(S, ' , '),
 ppsem(latex(S), Psi),
 write(S, ')').

ppsem(user, [lmd, X, Phi]) :- (!), write('L'),
 write(X),
 ppsem(user, Phi).

ppsem(latex(S), [lmd, X, Phi]) :- (!), write(S, '\lambda '),
 write(S, X),
 ppsem(latex(S), Phi).

ppsem(latex(S), [mapPhin, N, J]) :- (!), write(S, '({\Phi^n}^+\ '),
      ppsem(latex(S), N),
      write(S, '\ '),
      ppsem(latex(S), J),
```

```
      write(S, `)`).

ppsem(user, [fst, Phi]) :- (!), write('pi1'), ppsem(user, Phi).

ppsem(latex(S),[fst, Phi]) :- (!), write(S, `\pi_1`), ppsem(latex(S), Phi).

ppsem(user, [snd, Phi]) :- (!), write('pi2'), ppsem(user,Phi).

ppsem(latex(S), [snd, Phi]) :- (!), write(S, `\pi_2`), ppsem(latex(S), Phi).

ppsem(user, [1, Phi]) :- (!), write('iota1'), ppsem(user, Phi).

ppsem(latex(S),[1, Phi]) :- (!), write(S, `\iota_1`), ppsem(latex(S), Phi).

ppsem(user, [2, Phi]) :- (!), write('iota2'), ppsem(user,Phi).

ppsem(latex(S), [2, Phi]) :- (!), write(S, `\iota_2`), ppsem(latex(S), Phi).

ppsem(user, [iota, X, Phi]) :- (!), write('i'),
write(X),
pp(user, Phi).

ppsem(latex(S), [iota, X, Phi]) :- (!), write(S, `\iota`),
write(S, X),
ppsem(latex(S), Phi).

ppsem(latex(S), iota) :- (!), write(S, `\iota `).

ppsem(user, [xst, X, Phi]) :- (!), write('E'),
write(X),
ppsem(user, Phi).

ppsem(latex(S), [xst, X, Phi]) :- (!), write(S, `\exists `),
write(S, X),
ppsem(latex(S), Phi).

ppsem(user, [all, X, Phi]) :- (!), write('A'),
write(X),
ppsem(user, Phi).

ppsem(latex(S), [all, X, Phi]) :- (!), write(S, `\forall `),
write(S, X),
ppsem(latex(S), Phi).

ppsem(user, [xst2, X, Phi]) :- (!), write('E^2'),
write(X),
ppsem(user, Phi).

ppsem(latex(S), [xst2, X, Phi]) :- (!), write(S, `\exists^2`),
write(S, X),
ppsem(S, Phi).

ppsem(user, [up, Phi]) :- (!), write('^'),
ppsem(user, Phi).

ppsem(latex(S), [up, Phi]) :- (!), write(S, `\mbox{\^{}}`),
ppsem(latex(S), Phi).

ppsem(user, [dn, Phi]) :- (!), write('v'),
ppsem(user, Phi).

ppsem(latex(S), [dn, Phi]) :- (!), write(S, `\mbox{\v{}}`),
ppsem(latex(S), Phi).

ppsem(user, [or, Phi, Psi]) :- (!), write('['),
ppsem(user, Phi),
write(' v '),
ppsem(user, Psi),
write(']').
```

```
ppsem(latex(S), [or, Phi, Psi]) :- !, write(S, '['),
ppsem(latex(S), Phi),
write(S, '\vee '),
ppsem(latex(S), Psi),
write(S, ']').

ppsem(user, [and, Phi, Psi]) :- !, write('['),
ppsem(user, Phi),
write(' & '),
ppsem(user, Psi),
write(']').

ppsem(latex(S), [and, Phi, Psi]) :- !, write(S, '['),
ppsem(latex(S), Phi),
write(S, '\wedge '),
ppsem(latex(S), Psi),
write(S, ']').

ppsem(user, [imply, Phi, Psi]) :- !, write('['),
ppsem(user, Phi),
write(' -> '),
ppsem(user, Psi),
write(']').

ppsem(latex(S), [imply, Phi, Psi]) :- !, write(S, '['),
ppsem(latex(S), Phi),
write(S, '\rightarrow '),
ppsem(latex(S), Psi),
write(S, ']').

ppsem(user, [eq, Phi, Psi]) :- !, write('['),
ppsem(user, Phi),
write(' = '),
ppsem(user, Psi),
write(']').

ppsem(latex(S), [eq, Phi, Psi]) :- !, write(S, '['),
ppsem(latex(S), Phi),
write(S, '='),
ppsem(latex(S), Psi),
write(S, ']').

ppsem(user, [more_than, Phi, Psi]) :- !, write('['),
ppsem(user, Phi),
write(' > '),
ppsem(user, Psi),
write(']').

ppsem(latex(S), [more_than, Phi, Psi]) :- !, write(S, '['),
ppsem(latex(S), Phi),
write(S, '>'),
ppsem(latex(S), Psi),
write(S, ']').

ppsem(user, [group, Phi, Psi]) :- !, write('['),
ppsem(user, Phi),
write(' grp '),
ppsem(user, Psi),
write(']').

ppsem(latex(S), [group, Phi, Psi]) :- !, write(S, '['),
ppsem(latex(S), Phi),
write(S, ' grp '),
ppsem(latex(S), Psi),
write(S, ']').

ppsem(user, [plusubset, Phi, Psi]) :- !, write('['),
ppsem(user, Phi),
write(' pss '),
ppsem(user, Psi),
```

```
write(']').

ppsem(latex(S), [plusubset, Phi, Psi]) :- (!), write(S, '['),
ppsem(latex(S), Phi),
write(S, ' pss '),
ppsem(latex(S), Psi),
write(S, ']').

ppsem(user, [card, Phi]) :- (!), write('|'),
ppsem(user, Phi),
write('|').

ppsem(latex(S), [card, Phi]) :- (!), write(S, '|'),
ppsem(latex(S), Phi),
write(S, '|').

ppsem(user, [when, Phi, Psi]) :- (!), write('['),
ppsem(user, Psi),
write(' when '),
ppsem(user, Phi),
write(']').

ppsem(latex(S), [when, Phi, Psi]) :- (!), write(S, '['),
ppsem(latex(S), Psi),
write(' when '),
ppsem(latex(S), Phi),
write(S, ']').

ppsem(user, [not, Phi]) :- (!), write('-'),
ppsem(user, Phi).

ppsem(latex(S), [not, Phi]) :- (!), write(S, `\neg `),
ppsem(latex(S), Phi).

ppsem(user, Phi) :- write(Phi).

ppsem(latex(Stream), Phi) :- write(Stream, `{\it `),
      write(Stream, Phi),
      write(Stream, '}').

% eval(+Phi, -NF) means that NF is the result of normalising the
% semantic form Phi

eval(Phi, NF) :-
numbervars(Phi, 0, _),
eval1(Phi, NF).

% eval1(+Phi, -NF) means that NF is the result of normalising
% the frozen semantic form Phi

eval1(Phi, NF) :-
contract(Phi, Phi1), (!),
eval1(Phi1, NF).

eval1(Phi, Phi).

% contract(+Phi, -Phi1) means that Phi1 is the result of applying one
% contraction step to the frozen semantic form Phi

contract([app, [lmd, X, Phi], Psi], Chi) :-
subst(Psi, X, Phi, Chi).

contract([fst, [pair, X, _]], X).
contract([snd, [pair, _, Y]], Y).

contract([case, [1, Chi], X, Phi, _, _], Omega) :-
subst(Chi, X, Phi, Omega).

contract([case, [2, Chi], _, _, Y, Psi], Omega) :-
subst(Chi, Y, Psi, Omega).
```

```
contract([dn, [up, Phi]], Phi).

contract([H|T], [H|T1]) :-
contractlist(T, T1).

contract([mapapply, [X], Z], [[app, X, Z]]).

contract([mapapply, [X, Y|L], Z], [[app, X, Z]|[mapapply, [Y|L], Z]]).

contract([app, [app, [mapPhin, 0, J], Z], [X]], [J, X, Z]).

contract([app, [app, [mapPhin, 0, J], Z], [X, Y|L]],
     [J, X, [app, [app, [mapPhin, 0, J], Z], [Y|L]]]).

contract([app, [app, [app, [mapPhin, [app, s, N], J], Y], L], Z],
     [app, [app, [mapPhin, N, J], [app, Y, Z]], [mapapply, L, Z]]).

contract([eq, Phi, Phi], true).

contract([and, Phi, true], Phi).
contract([and, true, Phi], Phi).

contractlist([Phi|Phis], [Phi1|Phis]) :-
contract(Phi, Phi1).

contractlist([Phi|Phis], [Phi|Phis1]) :-
contractlist(Phis, Phis1).

% subst(+Phi, +X, +Psi, -NPsi) means that NPsi is the result of replacing
% by Phi all Xs in the frozen semantic form Psi

subst(Phi, X, X, Phi) :- (!).

subst(_, _, C, C) :-
atom(C), (!).

subst(_, _, C, C) :-
integer(C), (!).

subst(_, _, X, X) :-
X = '$VAR'(_), (!).

subst(Phi, X, [H|T], [H1|T1]) :-
subst(Phi, X, H, H1),
subst(Phi, X, T, T1).

% pppros(+Out, +Pros) pretty prints prosodic form Pros.

pppros(user, [b(Ws)]) :- (!),
     write('['),
     pppros(user, Ws),
     write(']').

pppros(user, [b(Ws)|Ws2]) :- (!),
     write('['),
     pppros(user, Ws),
     write(']'),
     write('+'),
     pppros(user, Ws2).

pppros(user, [W]) :- (!),
     write(W).

pppros(user, [W1, W2|Ws]) :- (!),
     write(W1),
     write('+'),
     pppros(user, [W2|Ws]).

pppros(user, Alpha) :-
```

```
      write(Alpha).

pppros(latex(S), [b(Ws)]) :- (!),
      write(S, '['),
      pppros(latex(S), Ws),
      write(S, ']').

pppros(latex(S), [b(Ws)|Ws2]) :- (!),
      write(S, '['),
      pppros(latex(S), Ws),
      write(S, ']'),
      write(S, '{+}'),
      pppros(latex(S), Ws2).

pppros(latex(S), [W]) :- (!),
      write(S, `{\bf `),
      write(S, W),
      write(S, '}').

pppros(latex(S), [W1, W2|Ws]) :- (!),
      write(S, `{\bf `),
      write(S, W1),
      write(S, '}'),
      write(S, '{+}'),
      pppros(latex(S), [W2|Ws]).

pppros(latex(S), Alpha) :-
      write(S, Alpha).

% pptype(+Out, +A) pretty prints type A.

pptype(Out, A) :- (!), pptype(Out, outer, A).

% pptype(+Out, +M, +A) pretty prints type A ommitting outer parentheses
% if M IS `outer' but not if it is `inner'.

pptype(user, _, n(G)) :- (!), write('N'), write(G).
pptype(user, _, s(F)) :- (!), write('S'), write(F).
pptype(user, _, cp(C)) :- (!), write('CP'), write(C).
pptype(user, _, pp(F)) :- (!), write('PP'), write(F).
pptype(user, _, cn(G)) :- (!), write('CN'), write(G).
pptype(user, _, si) :- (!), write('Si').
pptype(user, _, q) :- (!), write('Q').

pptype(user, _, w(W)) :- (!), write('W'), write(W). % 1

pptype(user, M, A bs C) :- (!),                     % 2
    doopenparen(user, M, A),
    write('\'),
    docloseparen(user, M, C).

pptype(user, M, C/B) :- (!),                        % 3
    doopenparen(user, M, C),
    write('/'),
    docloseparen(user, M, B).

pptype(user, M, A*B) :- (!),                        % 4
    doopenparen(user, M, A),
    write('*'),
    docloseparen(user, M, B).

pptype(user, _, i) :- (!), write('I').              % 5

pptype(user, M, A in C) :- (!),                     % 6+
    doopenparen(user, M, A),
    write('in'),
    docloseparen(user, M, C).

pptype(user, M, A inn C) :- (!),                    % 6-
    doopenparen(user, M, A),
```

```
    write('inn'),
    docloseparen(user, M, C).

pptype(user, M, C ci B) :- (!),                    % 7+
    doopenparen(user, M, C),
    write('ci'),
    docloseparen(user, M, B).

pptype(user, M, C cin B) :- (!),                   % 7-
    doopenparen(user, M, C),
    write('cin'),
    docloseparen(user, M, B).

pptype(user, M, A dp B) :- (!),                    % 8+
    doopenparen(user, M, A),
    write('dp'),
    docloseparen(user, M, B).

pptype(user, M, A dpn B) :- (!),                   % 8-
    doopenparen(user, M, A),
    write('dpn'),
    docloseparen(user, M, B).

pptype(user, _, j) :- (!), write('J').             % 9

pptype(user, M, A&B) :- (!),                       % 10
    doopenparen(user, M, A),
    write('&'),
    docloseparen(user, M, B).

pptype(user, M, A+B) :- (!),                       % 11
    doopenparen(user, M, A),
    write('+'),
    docloseparen(user, M, B).

pptype(user, _, X u A) :- (!),                     % 12
    write('AA'),
    write(X),
    pptype(user, inner, A).

pptype(user, _, X e A) :- (!),                     % 13
    write('AE'),
    write(X),
    pptype(user, inner, A).

pptype(user, _, 'L'A) :- (!),                      % 14
    write('L'),
    pptype(user, inner, A).

pptype(user, _, 'M'A) :- (!),                      % 15
    write('M'),
    pptype(user, inner, A).

pptype(user, _, ab A) :- (!),                      % 16
    write('[]-1'),
    pptype(user, inner, A).

pptype(user, _, br A) :- (!),                      % 17
    write('<>'),
    pptype(user, inner, A).

pptype(user, _, '!'A) :- (!),                      % 18
    write('!'),
    pptype(user, inner, A).

pptype(user, _, '?'A) :- (!),                      % 19
    write('?'),
    pptype(user, inner, A).

pptype(user, M, A lca B) :- (!),                   % 20
```

```
    doopenparen(user, M, A),
    write(' lca '),
    docloseparen(user, M, B).

pptype(user, M, A-B) :- (!),                    % 21
    doopenparen(user, M, A),
    write('-'),
    docloseparen(user, M, B).

% pptype(user, _, -A) :- (!),
%     write('-'),
%     pptype(user, inner, A).

pptype(user, M, A liu C) :- (!),                % 22
    doopenparen(user, M, A),
    write('-o'),
    docloseparen(user, M, C).

pptype(user, M, C lio B) :- (!),                % 23
    doopenparen(user, M, B),
    write('*-'),
    docloseparen(user, M, C).

pptype(user, M, A lip B) :- (!),                % 24
    doopenparen(user, M, A),
    write('.o'),
    docloseparen(user, M, B).

pptype(user, M, A riu C) :- (!),                % 25
    doopenparen(user, M, A),
    write('-*'),
    docloseparen(user, M, C).

pptype(user, M, C rio B) :- (!),                % 26
    doopenparen(user, M, C),
    write('o-'),
    docloseparen(user, M, B).

pptype(user, M, A rip B) :- (!),                % 27
    doopenparen(user, M, A),
    write('o.'),
    docloseparen(user, M, B).

pptype(user, M, A uii C) :- (!),                % 28+
    doopenparen(user, M, A),
    write('uii'),
    docloseparen(user, M, C).

pptype(user, M, A uiin C) :- (!),               % 28-
    doopenparen(user, M, A),
    write('uiin'),
    docloseparen(user, M, C).

pptype(user, M, C uic B) :- (!),                % 29+
    doopenparen(user, M, C),
    write('uic'),
    docloseparen(user, M, B).

pptype(user, M, C uicn B) :- (!),               % 29+
    doopenparen(user, M, C),
    write('uicn'),
    docloseparen(user, M, B).

pptype(user, M, A uidp B) :- (!),               % 30+
    doopenparen(user, M, A),
    write('uidp'),
    docloseparen(user, M, B).

pptype(user, M, A uidpn B) :- (!),              % 30-
    doopenparen(user, M, A),
```

```
    write('uidpm'),
    docloseparen(user, M, B).

pptype(user, M, A lii C) :- (!),                    % 31+
    doopenparen(user, M, A),
    write('lii'),
    docloseparen(user, M, C).

pptype(user, M, A liin C) :- (!),                   % 31-
    doopenparen(user, M, A),
    write('liin'),
    docloseparen(user, M, C).

pptype(user, M, C lic B) :- (!),                    % 32+
    doopenparen(user, M, B),
    write('lic'),
    docloseparen(user, M, C).

pptype(user, M, C licn B) :- (!),                   % 32-
    doopenparen(user, M, C),
    write('lic'),
    docloseparen(user, M, B).

pptype(user, M, A lidp B) :- (!),                   % 33+
    doopenparen(user, M, A),
    write('lidp'),
    docloseparen(user, M, B).

pptype(user, M, A lidpn B) :- (!),                  % 33-
    doopenparen(user, M, A),
    write('lidpn'),
    docloseparen(user, M, B).

pptype(user, M, A iac B) :- (!),                    % 34
    doopenparen(user, M, A),
    write('|&|'),
    docloseparen(user, M, B).

pptype(user, M, A iad B) :- (!),                    % 35
    doopenparen(user, M, A),
    write('|+|'),
    docloseparen(user, M, B).

pptype(user, _, X iu A) :- (!),                     % 36
    write('A'),
    write(X),
    pptype(user, inner, A).

pptype(user, _, X ie A) :- (!),                     % 37
    write('E'),
    write(X),
    pptype(user, inner, A).

pptype(user, _, iL A) :- (!),                       % 38
    write('iL'),
    pptype(user, inner, A).

pptype(user, _, iM A) :- (!),                       % 39
    write('iM'),
    pptype(user, inner, A).

pptype(user, _, lp A) :- (!),                       % 40
    write('<|-1'),
    pptype(user, inner, A).

pptype(user, _, li A) :- (!),                       % 41
    write('<|'),
    pptype(user, inner, A).

pptype(user, _, rp A) :- (!),                       % 42
```

```
    write('|>-1'),
    pptype(user, inner, A).

pptype(user, _, ri A) :- (!),                       % 43
    write('|>'),
    pptype(user, inner, A).

pptype(user, _, sp A) :- (!),                       % 44+
    write('v'),
    pptype(user, inner, A).

pptype(user, _, spn A) :- (!),                      % 44-
    write('v-'),
    pptype(user, inner, A).

pptype(user, _, bg A) :- (!),                       % 45+
    write('^'),
    pptype(user, inner, A).

pptype(user, _, bgn A) :- (!),                      % 45-
    write('^-'),
    pptype(user, inner, A).

pptype(user, M, C nd B) :- (!),                     % 46
    doopenparen(user, M, C),
    write('nd'),
    docloseparen(user, M, B).

pptype(user, M, A np B) :- (!),                     % 47
    doopenparen(user, M, A),
    write('np'),
    docloseparen(user, M, B).

pptype(user, M, A nin C) :- (!),                    % 48
    doopenparen(user, M, A),
    write('nin'),
    docloseparen(user, M, C).

pptype(user, M, C nci B) :- (!),                    % 49
    doopenparen(user, M, C),
    write('nci'),
    docloseparen(user, M, B).

pptype(user, M, A ndp B) :- (!),                    % 50
    doopenparen(user, M, A),
    write('nwrap'),
    docloseparen(user, M, B).

% pptype(user, M, A < B) :- (!),
%     doopenparen(user, M, A),
%     write('<'),
%     docloseparen(user, M, B).

% pptype(user, _, forall2(X, L, A)) :- (!),
%       write('A2'),
%       write(X),
%       write(':{'),
%       pptypelst(user, L),
%       write('}'),
%       pptype(user, outer, A).

pptype(latex(S), _, n(G)) :- (!), write(S, 'N'), write(S, G).
pptype(latex(S), _, s(F)) :- (!), write(S, 'S'), write(S, F).
pptype(latex(S), _, cp(C)) :- (!), write(S, `{\it CP}`), write(S, C).
pptype(latex(S), _, pp(F)) :- (!), write(S, `{\it PP}`), write(S, `{\it `),
      write(S, F), write(S, '}').
pptype(latex(S), _, cn(G)) :- (!), write(S, `{\it CN}`), write(S, `{\it `),
      write(S, G), write(S, '}').
pptype(latex(S), _, si) :- (!), write(S, 'Si').
pptype(latex(S), _, q) :- (!), write(S, 'Q').
```

```
pptype(latex(S), _, w(W)) :- !, write(S, 'W'), write(S, W). % 1

pptype(latex(S), M, A bs C) :- !,                          % 2
    doopenparen(latex(S), M, A),
    write(S, '\backslash '),
    doclosepareN(latex(S), M, C).

pptype(latex(S), M, C/B) :- !,                             % 3
    doopenparen(latex(S), M, C),
    write(S, '/'),
    doclosepareN(latex(S), M, B).

pptype(latex(S), M, A*B) :- !,                             % 4
    doopenparen(latex(S), M, A),
    write(S, '{\bullet}'),
    doclosepareN(latex(S), M, B).

pptype(latex(S), _, i) :- !, write(S, 'I').                % 5

pptype(latex(S), M, A in C) :- !,                          % 6+
    doopenparen(latex(S), M, A),
    write(S, '{\stackrel{\downarrow}{}}'),
    doclosepareN(latex(S), M, C).

pptype(latex(S), M, A inn C) :- !,                         % 6-
    doopenparen(latex(S), M, A),
    write(S, '{\stackrel{\downarrow}{_{-}}}'),
    doclosepareN(latex(S), M, C).

pptype(latex(S), M, C ci B) :- !,                          % 7+
    doopenparen(latex(S), M, C),
    write(S, '{\stackrel{\uparrow}{}}'),
    doclosepareN(latex(S), M, B).

pptype(latex(S), M, C cin B) :- !,                         % 7-
    doopenparen(latex(S), M, C),
    write(S, '{\stackrel{\uparrow}{_{-}}}'),
    doclosepareN(latex(S), M, B).

pptype(latex(S), M, A dp B) :- !,                          % 8+
    doopenparen(latex(S), M, A),
    write(S, '{\stackrel{\odot}{}}'),
    doclosepareN(latex(S), M, B).

pptype(latex(S), M, A dpn B) :- !,                         % 8-
    doopenparen(latex(S), M, A),
    write(S, '{\stackrel{\odot}{_{-}}}'),
    doclosepareN(latex(S), M, B).

pptype(latex(S), _, j) :- !, write(S, 'J').                % 9

pptype(latex(S), M, A&B) :- !,                             % 10
    doopenparen(latex(S), M, A),
    write(S, '{\&}'),
    doclosepareN(latex(S), M, B).

pptype(latex(S), M, A+B) :- !,                             % 11
    doopenparen(latex(S), M, A),
    write(S, '{\oplus}'),
    doclosepareN(latex(S), M, B).

pptype(latex(S), _, X u A) :- !,                           % 12
    write(S, '{\bigwedge}'),
    write(S, X),
    pptype(latex(S), inner, A).

pptype(latex(S), _, X e A) :- !,                           % 13
    write(S, '{\bigvee}'),
    write(S, X),
```

```
      pptype(latex(S), inner, A).

pptype(latex(S), _, 'L'A) :- (!),                          % 14
    write(S, `{\square}`),
    pptype(latex(S), inner, A).

pptype(latex(S), _, 'M'A) :- (!),                          % 15
    write(S, `{\Diamond}`),
    pptype(latex(S), inner, A).

pptype(latex(S), _, ab A) :- (!),                          % 16
    write(S, `{[]^{-1}}`),
    pptype(latex(S), inner, A).

pptype(latex(S), _, br A) :- (!),                          % 17
    write(S, `{\langle\rangle}`),
    pptype(latex(S), inner, A).

pptype(latex(S), _, '!'A) :- (!),                          % 18
    write(S, '!'),
    pptype(latex(S), inner, A).

pptype(latex(S), _, '?'A) :- (!),                          % 19
    write(S, '?'),
    pptype(latex(S), inner, A).

pptype(latex(S), M, B lca A) :- (!),                       % 20
    doopenparen(latex(S), M, B),
    write(S, '{|}'),
    docloseparen(latex(S), M, A).

pptype(latex(S), M, A-B) :- (!),                           % 21
    doopenparen(latex(S), M, A),
    write(S, '{-}'),
    docloseparen(latex(S), M, B).

pptype(latex(S), M, A liu C) :- (!),                       % 22
    doopenparen(latex(S), M, A),
    write(S, `{\multimap}`),
    docloseparen(latex(S), M, C).

pptype(latex(S), M, C lio B) :- (!),                       % 23
    doopenparen(latex(S), M, B),
    write(S, `{\multimapdotinv}`),
    docloseparen(latex(S), M, C).

pptype(latex(S), M, A lip B) :- (!),                       % 24
    doopenparen(latex(S), M, A),
    write(S, `{\LEFTcircle}`),
    docloseparen(latex(S), M, B).

pptype(latex(S), M, A riu C) :- (!),                       % 25
    doopenparen(latex(S), M, A),
    write(S, `{\multimapdot}`),
    docloseparen(latex(S), M, C).

pptype(latex(S), M, C rio B) :- (!),                       % 26
    doopenparen(latex(S), M, C),
    write(S, `{\multimapinv}`),
    docloseparen(latex(S), M, B).

pptype(latex(S), M, A rip B) :- (!),                       % 27
    doopenparen(latex(S), M, A),
    write(S, `{\RIGHTcircle}`),
    docloseparen(latex(S), M, B).

pptype(latex(S), M, A uii C) :- (!),                       % 28+
    doopenparen(latex(S), M, A),
    write(S, `{\stackrel{%\begin{picture}(7,8)(0,-2)\put(0.7,7){\rotatebox{-90}{$\multimap$}}\end{picture}}{_{}}}`),
    docloseparen(latex(S), M, C).
```

```
pptype(latex(S), M, A uiin C) :- (1),                                   % 28-
    doopenparen(latex(S), M, A),
    write(S, '{\stackrel{\begin{picture}(7,8)(0,-2)\put(0.7,7){\rotatebox{-90}{$\multimap$}}\end{picture}}{_{_-}}}'),
    docloseparen(latex(S), M, C).

pptype(latex(S), M, C uic B) :- (1),                                    % 29+
    doopenparen(latex(S), M, C),
    write(S, '{\stackrel{\begin{picture}(7,7)(0,-2)\put(1.5,-3){\rotatebox{90}{$\multimapdot$}}\end{picture}}{_{}}}'),
    docloseparen(latex(S), M, B).

pptype(latex(S), M, C uicn B) :- (1),                                   % 29-
    doopenparen(latex(S), M, C),
    write(S, '{\stackrel{\begin{picture}(7,7)(0,-2)\put(1.5,-3){\rotatebox{90}{$\multimapdot$}}\end{picture}}{_{_-}}}'),
    docloseparen(latex(S), M, B).

pptype(latex(S), M, A uidp B) :- (1),                                   % 30+
    doopenparen(latex(S), M, A),
    write(S, '{\stackrel{\begin{picture}(7,7)(0,0)\put(0,5.5){\rotatebox{-90}{$\LEFTcircle$}}\end{picture}}{_{}}}'),
    docloseparen(latex(S), M, B).

pptype(latex(S), M, A uidpn B) :- (1),                                  % 30-
    doopenparen(latex(S), M, A),
    write(S, '{\stackrel{\begin{picture}(7,7)(0,0)\put(0,5.5){\rotatebox{-90}{$\LEFTcircle$}}\end{picture}}{_{_-}}}'),
    docloseparen(latex(S), M, B).

pptype(latex(S), M, A lii C) :- (1),                                    % 31+
    doopenparen(latex(S), M, A),
    write(S, '{\stackrel{\begin{picture}(7,8)(0,-2)\put(0.7,7){\rotatebox{-90}{$\multimapdot$}}\end{picture}}{_{}}}'),
    docloseparen(latex(S), M, C).

pptype(latex(S), M, A liin C) :- (1),                                   % 31-
    doopenparen(latex(S), M, A),
    write(S, '{\stackrel{\begin{picture}(7,8)(0,-2)\put(0.7,7){\rotatebox{-90}{$\multimapdot$}}\end{picture}}{_{_-}}}'),
    docloseparen(latex(S), M, C).

pptype(latex(S), M, C lic B) :- (1),                                    % 32+
    doopenparen(latex(S), M, B),
    write(S, '{\stackrel{\begin{picture}(7,7)(0,-2)\put(1.5,-3){\rotatebox{90}{$\multimap$}}\end{picture}}{_{}}}'),
    docloseparen(latex(S), M, C).

pptype(latex(S), M, C licn B) :- (1),                                   % 32-
    doopenparen(latex(S), M, C),
    write(S, '{\stackrel{\begin{picture}(7,7)(0,-2)\put(1.5,-3){\rotatebox{90}{$\multimap$}}\end{picture}}{_{_-}}}'),
    docloseparen(latex(S), M, B).

pptype(latex(S), M, A lidp B) :- (1),                                   % 33+
    doopenparen(latex(S), M, A),
    write(S, '{\stackrel{\begin{picture}(7,7)(0,0)\put(0,5.5){\rotatebox{-90}{$\RIGHTcircle$}}\end{picture}}{_{}}}'),
    docloseparen(latex(S), M, B).

pptype(latex(S), M, A lidpn B) :- (1),                                  % 33-
    doopenparen(latex(S), M, A),
    write(S, '{\stackrel{\begin{picture}(7,7)(0,0)\put(0,5.5){\rotatebox{-90}{$\RIGHTcircle$}}\end{picture}}{_{_-}}}'),
    docloseparen(latex(S), M, B).

pptype(latex(S), M, A iac B) :- (1),                                    % 34
    doopenparen(latex(S), M, A),
    write(S, '{\sqcap}'),
    docloseparen(latex(S), M, B).

pptype(latex(S), M, A iau B) :- (1),                                    % 35
    doopenparen(latex(S), M, A),
    write(S, '{\sqcup}'),
    docloseparen(latex(S), M, B).

pptype(latex(S), _, X iu A) :- (1),                                     % 36
    write(S, '{\forall}'),
    write(S, X),
    pptype(latex(S), inner, A).
```

```
pptype(latex(S), _, X ie A) :- (!),                          % 37
      write(S, `{\exists}`),
      write(S, X),
      pptype(latex(S), inner, A).

pptype(latex(S), _, iL A) :- (!),                            % 38
    write(S, `{\blacksquare}`),
    pptype(latex(S), inner, A).

pptype(latex(S), _, iM A) :- (!),                            % 39
    write(S, `{\blacklozenge}`),
    pptype(latex(S), inner, A).

pptype(latex(S), _, lp A) :- (!),                            % 40
    write(S, `\mbox{$\lhd^{-1}$}`),
    pptype(latex(S), inner, A).

pptype(latex(S), _, li A) :- (!),                            % 41
    write(S, `\mbox{$\lhd$}`),
    pptype(latex(S), inner, A).

pptype(latex(S), _, rp A) :- (!),                            % 42
    write(S, `\mbox{$\rhd^{-1}$}`),
    pptype(latex(S), inner, A).

pptype(latex(S), _, ri A) :- (!),                            % 43
    write(S, `\mbox{$\rhd$}`),
    pptype(latex(S), inner, A).

pptype(latex(S), _, sp A) :- (!),                            % 44+
    write(S, `\mbox{$\check{\ }$}`),
    pptype(latex(S), inner, A).

pptype(latex(S), _, spn A) :- (!),                           % 44-
    write(S, `\mbox{$\check{_{-}}$}`),
    pptype(latex(S), inner, A).

pptype(latex(S), _, bg A) :- (!),                            % 45+
    write(S, `\mbox{$\hat{\ }$}`),
    pptype(latex(S), inner, A).

pptype(latex(S), _, bgn A) :- (!),                           % 45-
    write(S, `\mbox{$\hat{_{-}}$}`),
    pptype(latex(S), inner, A).

pptype(latex(S), M, C nd B) :- (!),                          % 46
    doopenparen(latex(S), M, C),
    write(S, `{\div}`),
    docloseparen(latex(S), M, B).

pptype(latex(S), M, A np B) :- (!),                          % 47
    doopenparen(latex(S), M, A),
    write(S, `{\times}`),
    docloseparen(latex(S), M, B).

pptype(latex(S), M, A nin C) :- (!),                         % 48
    doopenparen(latex(S), M, A),
    write(S, `{\Downarrow}`),
    docloseparen(latex(S), M, C).

pptype(latex(S), M, C nci B) :- (!),                         % 49
    doopenparen(latex(S), M, C),
    write(S, `{\Uparrow}`),
    docloseparen(latex(S), M, B).

pptype(latex(S), M, A ndp B) :- (!),                         % 50
    doopenparen(latex(S), M, A),
    write(S, `\mbox{\footnotesize\begin{picture}(0,0)\put(2.2, 1){$\circ$}\put(0, 1){$\bigcirc$}\end{picture}\ \ \ }`),
    docloseparen(latex(S), M, B).
```

```
% pptype(latex(S), _, -A) :- (!),
%     write(S, '\neg '),
%     pptype(latex(S), inner, A).

% pptype(latex(S), M, A < B) :- (!),
%     doopenparen(latex(S), M, A),
%     write(S, '{<}'),
%     docloseparen(latex(S), M, B).

% pptype(latex(S), _, forall2(X, L, A)) :- (!),
%     write(S, '{\forall^2}'),
%     write(S, X),
%     write(S, '\in\{'),
%     pptypelst(latex(S), L),
%     write(S, '\}'),
%     pptype(latex(S), outer, A).

pptype(user, _, A) :- write(A).

pptype(latex(Stream), _, A) :- write(Stream, A).

pptypelst(user, [A]) :- (!), pptype(user, outer, A).

pptypelst(user, [H|T]) :- pptype(user, outer, H),
    write(', '),
    pptypelst(user, T).

pptypelst(latex(S), [A]) :- (!), pptype(latex(S), outer, A).

pptypelst(latex(S), [H|T]) :- pptype(latex(S), outer, H),
    write(S, ' '),
    pptypelst(latex(S), T).

doopenparen(user, inner, A) :- write('('),pptype(user, inner, A).
doopenparen(user, outer, A) :- pptype(user, inner, A).

doopenparen(latex(S), inner, A) :- write(S, '('),pptype(latex(S), inner, A).
doopenparen(latex(S), outer, A) :- pptype(latex(S), inner, A).

docloseparen(user, inner, A) :- pptype(user, inner, A), write(')').
docloseparen(user, outer, A) :- pptype(user, inner, A).

docloseparen(latex(S), inner, A) :- pptype(latex(S), inner, A), write(S, ')').
docloseparen(latex(S), outer, A) :- pptype(latex(S), inner, A).

pplex :-
    lex(Alpha, A, Phi),
    numbervars((A, Phi), 0, _),
    pppros(user, Alpha),
    write(': '),
    pptype(user, A),
    write(' : '),
    numbervars(Phi, 0, _),
    ppsem(user, Phi), nl,
    fail.

pplexlatex :-
    open('s.tex', write, S), nl(S),
    pplexlatex(S).

pplexlatex(S) :-
    nl(S), write(S, '$'),%\begin{array}{l}
    lex(Alpha, A, Phi),
    numbervars((A, Phi), 0, _),
    nl(S),
    pppros(latex(S), Alpha),
    write(S, ' : '),
    pptype(latex(S), A),
```

```
    write(S, ': '),
    numbervars(Phi, 0, _),
    ppsem(latex(S), Phi),
    write(S, '\\'),
    fail.

pplexlatex(S) :-
    write(S, '$'), nl(S),%\end{array}
    close(S).

% ppconfig(Out, +S, +Gamma) pretty prints the configuration Gamma
% including semantics if S is 'yes' but not if it is 'no'.  It writes
% to the user if Out is 'user' and writes to Stream if Out is
% latex(Stream).

ppconfig(_, _, []).

ppconfig(Out, S, [F]) :-
    ppfactor(Out, S, F).

ppconfig(user, S, [F1, F2|Gamma]) :-
    ppfactor(user, S, F1),
    write(', '),
    ppconfig(user, S, [F2|Gamma]).

ppconfig(latex(Stream), S, [F1, F2|Gamma]) :-
    ppfactor(latex(Stream), S, F1),
    write(Stream, ', '),
    ppconfig(latex(Stream), S, [F2|Gamma]).

ppfactor(user, _, 1) :-
    write('1').

ppfactor(latex(Stream), _, 1) :-
    write(Stream, '{\tt 1}').

ppfactor(Stream, S, l(A, Phi, Ls)) :-
    pptypefactor(Stream, S, A, Ls),
    dosem(Stream, S, Phi).

ppfactor(user, S, f(A, Phi, Ls)) :-
      pptypefactor(user, S, A, Ls),
      dosem(user, S, Phi).

ppfactor(latex(Stream), S, f(A, Phi, Ls)) :-
      write(Stream, '\mbox{\fbox{$'),
      pptypefactor(latex(Stream), S, A, Ls),
      dosem(latex(Stream), S, Phi),
      write(Stream, '$}}').

ppfactor(user, S, b([]: Gamma)) :-
      write('['),
      ppconfig(user, S, Gamma),
      write(']').

ppfactor(latex(Stream), S, b([]: Gamma)) :-
      write(Stream, '['),
      ppconfig(latex(Stream), S, Gamma),
      write(Stream, ']').

ppfactor(user, S, b([H|T]: Gamma)) :-
      write('['),
      ppconfig(user, S, [H|T]), write(';'),
      ppconfig(user, S, Gamma),
      write(']').

ppfactor(latex(Stream), S, b([]: Gamma)) :-
      write(Stream, '['),
      ppconfig(latex(Stream), S, Gamma),
      write(Stream, ']').
```

```
ppfactor(latex(Stream), S, b([H|T]: Gamma)) :-
      write(Stream, '['),
      ppconfig(latex(Stream), S, [H|T]), write(Stream, ';'),
      ppconfig(latex(Stream), S, Gamma),
      write(Stream, ']').

dosem(user, yes, Phi) :-
   write(': '),
   assert(copy(Phi)),
   retract(copy(Phi1)),
   numbervars(Phi1, 0, _),
   ppsem(user, Phi1).

dosem(latex(Stream), yes, Phi) :-
   write(Stream, ': '),
   assert(copy(Phi)),
   retract(copy(Phi1)),
   numbervars(Phi1, 0, _),
   ppsem(latex(Stream), Phi1).

dosem(_, no, _).

pptypefactor(Out, _, A, []) :-
   pptype(Out, outer, A).

pptypefactor(user, S, A, [L|Ls]) :-
   pptype(user, A),
   write('{'),
   ppconfiglst(user, S, [L|Ls]),
   write('}').

pptypefactor(latex(Stream), S, A, [L|Ls]) :-
   pptype(latex(Stream), A),
   write(Stream, '\{'),
   ppconfiglst(latex(Stream), S, [L|Ls]),
   write(Stream, '\}').

ppconfiglst(Out, S, [L]) :-
   ppconfig(Out, S, L).

ppconfiglst(user, S, [L1, L2|Ls]) :-
   ppconfig(user, S, L1),
   write(' : '),
   ppconfiglst(user, S, [L2|Ls]).

ppconfiglst(latex(Stream), S, [L1, L2|Ls]) :-
   ppconfig(latex(Stream), S, L1),
   write(Stream, ' : '),
   ppconfiglst(latex(Stream), S, [L2|Ls]).

% ppseq(+Out, +S, +Gamma, +A) pretty prints the sequent Gamma => A
% including semantics if S is 'yes' but not if it is 'no'. It writes
% to the user if Out is 'user' and writes to Stream if Out is
% latex(Stream).

ppseq(Out, S, Gamma, A) :- \+(\+((numbervars((Gamma, A), 0, _), ppseq2(Out, S, Gamma, A)))).

ppseq2(user, S, []: Gamma, A) :-
   ppconfig(user, S, Gamma),
   write(' => '),
   ppsucc(user, A).

ppseq2(user, S, [H|T]: Gamma, A) :-
   ppconfig(user, S, [H|T]),
   write(': '),
   ppconfig(user, S, Gamma),
   write(' => '),
   ppsucc(user, A).
```

```
ppseq2(latex(Stream), S, []: Gamma, A) :-
    ppconfig(latex(Stream), S, Gamma),
    write(Stream,`\ \Rightarrow\ `),
    ppsucc(latex(Stream), A).

ppseq2(latex(Stream), S, [H|T]: Gamma, A) :-
    ppconfig(latex(Stream), S, [H|T]),
    write(Stream, `;\ `),
    ppconfig(latex(Stream), S, Gamma),
    write(Stream,`\ \Rightarrow\ `),
    ppsucc(latex(Stream), A).

ppsucc(user, l(A)) :-
      pptype(user, outer, A).

ppsucc(user, f(A)) :-
      pptype(user, outer, A).

ppsucc(latex(Stream), l(A)) :-
      pptype(latex(Stream), outer, A).

ppsucc(latex(Stream), f(A)) :-
      write(Stream, `\fbox{$`),
      pptype(latex(Stream), outer, A),
      write(Stream, `$}`).

% ppprf(+Out +S, +N, Prf) pretty prints sequent proof Prf at indentation
% N including semantics if S is `yes' but not if it is `no'.  It writes
% to the user if Out is `user' and writes to Stream if Out is
% latex(Stream).

ppprf(user, S, N, prf(id, Gamma, A, [])) :- (!),
    dotab(N), ppseq(user, S, Gamma, A).

ppprf(user, S, N, prf(Rule, Gamma, A, SubPrfs)) :-
    dotab(N), ppseq(user, S, Gamma, A), write(' ['), writerule(user, Rule), write(']'),
    pppprflst(user, S, s(s(s(s(N)))), SubPrfs).

ppprf(latex(Stream), S, _, prf(id, Gamma, A, [])) :- (!),
    nl(Stream),
    write(Stream, `\prooftree`),
    nl(Stream),
    write(Stream, `\justifies`),
    nl(Stream),
    ppseq(latex(Stream), S, Gamma, A),
    nl(Stream),
    write(Stream, `\endprooftree`).

ppprf(latex(Stream), S, _, prf(Rule, Gamma, A, SubPrfs)) :-
    nl(Stream),
    write(Stream, `\prooftree`),
    pppprflst(latex(Stream), S, _, SubPrfs),
    nl(Stream),
    write(Stream, `\justifies`),
    nl(Stream),
    ppseq(latex(Stream), S, Gamma, A),
    nl(Stream),
    write(Stream, `\using `),
    writerule(latex(Stream), Rule),
    nl(Stream),
    write(Stream, `\endprooftree`).

pppprflst(_, _, _, []).

pppprflst(user, S, N, [H|T]) :-
    nl, ppprf(user,S, N, H),
    pppprflst(user, S, N, T).

pppprflst(latex(Stream), S, N, [H|T]) :-
    ppprf(latex(Stream),S, N, H),
```

```
    ppprflst(latex(Stream), S, N, T).

dotab(0).

dotab(s(N)) :- write(’ ’), dotab(N).

connwriting(words, ‘W‘, ‘{\it W}‘).                              % 1
connwriting(under, ‘\‘, ‘{\backslash}‘).                         % 2
connwriting(over, ‘/‘, ‘{/}‘).                                   % 3
connwriting(product, ‘*‘, ‘{\bullet}‘).                          % 4
connwriting(productunit, ‘I‘, ‘{\it I}‘).                        % 5
connwriting(infix, ‘in‘, ‘{\downarrow}‘).                        % 6+
connwriting(infixn, ‘inn‘, ‘{\stackrel{\downarrow}{_{_-}}}‘).   % 6-
connwriting(circumfix, ‘ci‘, ‘{\uparrow}‘).                      % 7+
connwriting(circumfixn, ‘cin‘, ‘{\stackrel{\uparrow}{_{_-}}}‘). % 7-
connwriting(dprod, ‘dp‘, ‘{\odot}‘).                             % 8+
connwriting(dprodn, ‘dpn‘, ‘{\stackrel{\odot}{_{_-}}}‘).        % 8-
connwriting(dproductunit, ‘J‘, ‘{\it J}‘).                       % 9
connwriting(aconj, ‘&‘, ‘{\&}‘).                                 % 10
connwriting(adisj, ‘+‘, ‘{\oplus}‘).                             % 11
connwriting(univq, ‘/\‘, ‘{\bigwedge}‘).                         % 12
connwriting(exstq, ‘\/‘, ‘{\bigvee}‘).                           % 13
connwriting(umod, ‘L‘, ‘{\Box}‘).                                % 14
connwriting(emod, ‘M‘, ‘{\diamond}‘).                            % 15
connwriting(abrack, ‘[]-1‘, ‘{[]^{-1}}‘).                     % 16
connwriting(brack, ‘<>‘, ‘{\langle\rangle}‘).                  % 17
connwriting(uexp, ‘!‘, ’{!}’).                                 % 18
connwriting(eexp, ‘?‘, ’{?}’).                                 % 19
connwriting(a, ‘|‘, ‘{|}‘).                                      % 20
connwriting(diff, ‘-‘, ‘{-}‘).                                   % 21
connwriting(liunder, ‘-o‘, ‘{\multimap}‘).                       % 22
connwriting(liover, ‘*-‘, ‘{\multimapdotinv}‘).                  % 23
connwriting(liproduct, ‘.o‘, ‘{\LEFTcircle}‘).                   % 24
connwriting(riunder, ‘-*‘, ‘{\multimapdot}‘).                    % 25
connwriting(riover, ‘o-‘, ‘{\multimapinv}‘).                     % 26
connwriting(riproduct, ‘o.‘, ‘{\RIGHTcircle}‘).                  % 27
connwriting(uiinfix, ‘uii‘, ‘{\stackrel{%\begin{picture}(7,8)(0,-2)\put(0.7,7){\rotatebox{-90}{$\multimap$}}\end{picture}}{_{}}}‘).                   % 28+
connwriting(uiinfixn, ‘uiin‘, ‘{\stackrel{%\begin{picture}(7,8)(0,-2)\put(0.7,7){\rotatebox{-90}{$\multimap$}}\end{picture}}{_{-}}}‘).                   % 28-
connwriting(uicircum, ‘uic‘, ‘{\stackrel{\begin{picture}(7,7)(0,-2)\put(1.5,-3){\rotatebox{90}{$\multimapdot$}}\end{picture}}{_{}}}‘).                  % 29+
connwriting(uicircumn, ‘uicn‘, ‘{\stackrel{\begin{picture}(7,7)(0,-2)\put(1.5,-3){\rotatebox{90}{$\multimapdot$}}\end{picture}}{_{-}}}‘).        % 29-
connwriting(uidprod, ‘uidp‘, ‘{\stackrel{\begin{picture}(7,7)(0,0)\put(0,5.5){\rotatebox{-90}{$\LEFTcircle$}}\end{picture}}{_{}}}‘).                  % 30+
connwriting(uidprodn, ‘uidpn‘, ‘{\stackrel{\begin{picture}(7,7)(0,0)\put(0,5.5){\rotatebox{-90}{$\LEFTcircle$}}\end{picture}}{_{-}}}‘).                 % 30-
connwriting(liinfix, ‘lii‘, ‘{\stackrel{\begin{picture}(7,8)(0,-2)\put(0.7,7){\rotatebox{-90}{$\multimapdot$}}\end{picture}}{_{}}}‘).                 % 31+
connwriting(liinfixn, ‘liin‘, ‘{\stackrel{\begin{picture}(7,8)(0,-2)\put(0.7,7){\rotatebox{-90}{$\multimapdot$}}\end{picture}}{_{-}}}‘).                % 31-
connwriting(licircum, ‘lic‘, ‘{\stackrel{\begin{picture}(7,7)(0,-2)\put(1.5,-3){\rotatebox{90}{$\multimap$}}\end{picture}}{_{}}}‘).                    % 32+
connwriting(licircumn, ‘licn‘, ‘{\stackrel{\begin{picture}(7,7)(0,-2)\put(1.5,-3){\rotatebox{90}{$\multimap$}}\end{picture}}{_{-}}}‘).          % 32-
connwriting(lidprod, ‘lidp‘, ‘{\stackrel{\begin{picture}(7,7)(0,0)\put(0,5.5){\rotatebox{-90}{$\RIGHTcircle$}}\end{picture}}{_{}}}‘).                 % 33+
connwriting(lidprodn, ‘lidpn‘, ‘{\stackrel{\begin{picture}(7,7)(0,0)\put(0,5.5){\rotatebox{-90}{$\RIGHTcircle$}}\end{picture}}{_{-}}}‘).                % 33-
connwriting(iaconj, ‘|&|‘, ‘{\sqcap}‘).                          % 34
connwriting(iadisj, ‘|+|‘, ‘{\sqcup}‘).                          % 35
connwriting(iunivq, ’A’, ‘{\forall}‘).                           % 36
connwriting(iexstq, ’E’, ‘{\exists}‘).                           % 37
connwriting(iumod, iL, ‘{\blacksquare}‘).                        % 38
connwriting(iemod, iM, ‘{\blacklozenge}‘).                       % 39
connwriting(lproj, ‘<|-1‘, ‘{\lhd^{-1}}‘).                       % 40
connwriting(linj, ‘<|‘, ‘{\lhd}‘).                               % 41
connwriting(rproj, ‘|>-1‘, ‘{\rhd^{-1}}‘).                       % 42
connwriting(rinj, ‘|>‘, ‘{\rhd}‘).                               % 43
connwriting(split, ‘v‘, ‘\mbox{$\check{\ }$}‘).                  % 44+
connwriting(splitn, ‘^-‘, ‘\mbox{$\check{_{_-}}$}‘).             % 44-
connwriting(bridge, ‘^‘, ‘\mbox{$\hat{\ }$}‘).                   % 45+
connwriting(bridgen, ‘^-‘, ‘\mbox{$\hat{_{_-}}$}‘).              % 45-
connwriting(ndiv, ‘-:-‘, ‘{\div}‘).                              % 46
connwriting(nprod, ‘X‘, ‘{\times}‘).                             % 47
connwriting(ninfix, ‘nin‘, ‘{\Downarrow}‘).                      % 48
connwriting(ncircumfix, ‘nci‘, ‘{\Uparrow}‘).                    % 49
connwriting(ndprod, ‘ndp‘, ‘\mbox{\footnotesize\begin{picture}(0,0)\put(2.2, 1){$\circ$}\put(0, 1){$\bigcirc$}\end{picture}\ \ \ }‘).                  % 50

writeruleuser(Op) :-
      connwriting(Op, T, _),
```

```
      write(T).

writerulelatex(S, Op) :-
      connwriting(Op, _, T),
      write(S, T).

writerule(user, left(Op)) :-
      writeruleuser(Op),
      write('L').

writerule(user, right(Op)) :-
      writeruleuser(Op),
      write('R').

writerule(user, perm(Op)) :-
      writeruleuser(Op),
      write('P').

writerule(user, contr(Op)) :-
      writeruleuser(Op),
      write('C').

writerule(user, expan(Op)) :-
      writeruleuser(Op),
      write('E').

writerule(latex(S), left(Op)) :-
      writerulelatex(S, Op),
      write(S, 'L').

writerule(latex(S), right(Op)) :-
      writerulelatex(S, Op),
      write(S, 'R').

writerule(latex(S), perm(Op)) :-
      writerulelatex(S, Op),
      write(S, 'P').

writerule(latex(S), contr(Op)) :-
      writerulelatex(S, Op),
      write(S, 'C').

writerule(latex(S), expan(Op)) :-
      writerulelatex(S, Op),
      write(S, 'E').

writerule(user, id) :- write('id').

writerule(latex(Stream), id) :- write(Stream, '{\it id}').

str(dwp('(7-7)'), [b([john]), walks], s(f)).
str(dwp('(7-16)'), [b([every, man]), talks], s(f)).
str(dwp('(7-19)'), [b([the, fish]), walks], s(f)).
str(dwp('(7-32)'), [b([every, man]), b([b([walks, or, talks])])], s(f)).
str(dwp('(7-34)'), [b([b([b([every, man]), walks, or, b([every, man]), talks])])], s(f)).
str(dwp('(7-39)'), [b([b([b([a, woman]), walks, and, b([she]), talks])])], s(f)).
str(dwp('(7-43, 45)'), [b([john]), believes, that, b([a, fish]), walks], s(f)).
str(dwp('(7-48, 49, 52)'), [b([every, man]), believes, that, b([a, fish]), walks], s(f)).
str(dwp('(7-57)'), [b([every, fish, such, that, b([it]), walks]), talks], s(f)).
str(dwp('(7-60, 62)'), [b([john]), seeks, a, unicorn], s(f)).
str(dwp('(7-73)'), [b([john]), is, bill], s(f)).
str(dwp('(7-76)'), [b([john]), is, a, man], s(f)).
str(dwp('(7-83)'), [necessarily, b([john]), walks], s(f)).
str(dwp('(7-86)'), [b([john]), walks, slowly], s(f)).
str(dwp('(7-91)'), [b([john]), tries, to, walk], s(f)).
%str(dwp('(7-94)'), [b([john]), tries, to, b([b([catch, a, fish, and, eat, it])])], s(f)).
str(dwp('(7-98)'), [b([john]), finds, a, unicorn], s(f)).
str(dwp('(7-105)'), [b([every, man, such, that, b([he]), loves, a, woman]), loses, her], s(f)).
str(dwp('(7-110)'), [b([john]), walks, in, a, park], s(f)).
str(dwp('(7-116, 118)'), [b([every, man]), doesnt, walk], s(f)).
```

```
str(1, [b([john]), walks], s(f)).
str(2, [b([john]), loves, mary], s(f)).
str(3, [b([john]), thinks, b([mary]), walks], s(f)).
str(4, [b([mary]), believes, b([john]), thinks, b([mary]), walks], s(f)).
str(5, [b([mary]), buys, john, coffee], s(f)).
str(6, [man, loved], cn(s(m))).
str(7, [b([john]), is, loved], s(f)).
str(8, [b([john]), is, loved, by, mary], s(f)).
str(9, [b([the, man]), walks], s(f)).
str(10, [b([the, man]), loves, the, woman], s(f)).
str(11, [man, b(b([that, walks]))]), cn(s(m))).
str(12, [b([the, man, b(b([that, walks]))]), talks], s(f)).
str(13, [b([the, man, b(b([that, loves, mary]))]), walks], s(f)).
str(14, [b([the, man, b(b([that, b([mary]), loves]))]), walks], s(f)).
str(15, [b([the, man, b(b([that, b([john]), thinks, b([mary]), loves]))]), walks], s(f)).
str(16, [b([the, horse]), raced, past, the, barn], s(f)).
str(17, [b([the, horse, raced, past, the, barn]), fell], s(f)).

str(18, [b([john]), walks, from, edinburgh], s(f)).
str(19, [b([the, man, from, edinburgh]), walks], s(f)).

str(20, [b([bond]), is, '007'], s(f)).
str(21, [b([bond]), is, teetotal], s(f)).
str(22, [b([bond]), is, b([b(['007', and, teetotal])])], s(f)).
str(23, [b([bond]), is, b([b([teetotal, and, '007'])])], s(f)).
str(24, [b([bond]), is, b([b([tall, and, '007'])])], s(f)).

str(tdc(43), [b([mary]), gave, the, man, the, cold, shoulder], s(f)).
str(tdc(4343), [b([mary]), gave, john, the, cold, shoulder], s(f)).
str(tdc(47), [b([john]), gave, every, book, to, mary], s(f)).
str(tdc(90), [b([mary]), thinks, b([someone]), left], s(f)).
str(tdc(53), [b([everyone]), loves, someone], s(f)).
str(tdc(54), [dog, b([b([that, b([mary]), saw, today])])], cn(s(n))).
str(tdc('58a'), [b([john]), slept, before, b([mary]), did], s(f)).
%str(tdc('58b'), [john, slept, and, mary, did, too], s(f)).
str(tdc(64), [mountain, b([b([the, painting, of, which, by, cezanne, b([john]), sold, for, tenmilliondollars])])], cn(s(n))).
%str(56, [thesis, the, height, of, the, lettering, on, the, first, line, of, the, second, page, of, the, third, chapter, of, which, is, '0.5cm'], cn(n)).
str(tdc(67), [b([john, b([b([who, jogs])])]), sneezed], s(f)).
str(tdc('70a'), [fortunately, b([john]), has, perseverance], s(f)).
str(tdc('70b'), [b([john]), fortunately, has, perseverance], s(f)).
str(tdc('70c'), [b([john]), has, fortunately, perseverance], s(f)).
str(tdc('70d'), [b([john]), has, perseverance, fortunately], s(f)).
str(tdc(73), [b([b([john]), studies, logic, and, b([charles]), phonetics])])], s(f)).
str(tdc(75), [b([john]), ate, more, donuts than, b([mary]), bought, bagels], s(f)).
str(tdc('86a'), [b([john]), bought, himself, coffee], s(f)).
%str(65, [dorothy, bet, the, straw, man, half, of, himself, that, she, would, reach, emerald, city, first], s).

str(w(1), [b([it]), rains], s(f)).
str(w(2), [b([mary]), gives, peter, the, cold, shoulder], s(f)).
str(w('3a'), [b([mary]), calls, peter, up], s(f)).
str(w('3b'), [b([mary]), calls, up, peter], s(f)).
str(w('4a'), [b([b([peter]), sees, mary, and, b([robin]), clark])])], s(f)).
str(w('4b'), [b([b([peter]), calls, up, mary, and, b([robin]), clark])])], s(f)).
str(w('4c'), [b([b([peter]), calls, mary, up, and, b([robin]), clark])])], s(f)).
str(w('4d'), [b([b([peter]), gives, mary, the, cold, shoulder, and, b([robin]), clark])])], s(f)).

str(rel(6), [man, b([b([that, walks])])], cn(s(m))).
str(rel(7), [man,b([b([that,b([mary]).likes])])], cn(s(m))).
%str(rel(8), [man, b([b([that, b([john]), thinks, that, b([mary]), believes, %that,
%b([the, girl])])])], cn(s(m))).
str(rel(9), [man,b([b([that,b([mary]).likes,today])])], cn(s(m))).
str(rel(10), [man, that, b([the, friends, of]), walk], cn(s(m))).
str(rel(11), [man, b([b([that, b([b([the, friends, of])]), admire])])], cn(s(m))).
str(rel(12), [paper, b([b([that, b([john]), filed, b([b([without, reading])])])])], cn(s(n))).
str(rel(13), [paper, b([b([that, b([b([the, editor, of])]), filed, b([b([without, reading])])])])], cn(s(n))).

str(crd(14), [b([b([john]), praises, mary, and, b([john]), laughs])])], s(f)).
str(crd(15), [b([john]), b([b([praises, mary, and, laughs])])], s(f)).
str(crd(16), [b([john]), b([b([likes, and, will, love])]), london], s(f)).
```

```
str(crd(17), [b([john]), b([b([gave, or, sent])]), mary, the, book], s(f)).
str(crd(18), [b([b([b([john]), or, b([mary])])]), sings], s(f)).
str(crd(19), [b([john]), loves, b([b([mary, and, himself])])], s(f)).
str(crd(20), [b([b([b([john]), likes, and, b([mary]), loves])]), london], s(f)).
str(crd(21), [b([john]), b([b([gave, the, book, and, sent, the, cd])]), to, mary], s(f)).
str(crd(22), [b([john]), gave, b([b([the, book, to, mary, and, the, cd, to, suzy])])], s(f)).
str(crd(23), [b([john]), saw, b([b([mary, today, and, bill, yesterday])])], s(f)).
%str(crd(24), [man, b([b([that, b([b([b([john]), saw, yesterday, and, b([bill]), saw, today])])])])], cn(s(m))).
str(crd(25), [man, b([b([that, b([john]), b([b([saw, yesterday, and, met, today])])])])], cn(s(m))).
str(crd(26), [b([b([b([john]), walks, b([mary]), talks, and, b([bill]), sings])])], s(f)).
str(crd(27), [b([john]), b([b([walks, talks, and, sings])])], s(f)).
str(crd(28), [b([john]), b([b([praises, likes, and, will, love])]), london], s(f)).

str(gen(one(1)), [in, the, beginning, b(['God']), created, b([b([the, heaven, and, the,
earth])])], s(f)).

%str(gen(one('2'(a))), ['And', b([b([b([the, earth]), was, b([b([without, form, and, void])])]), and, b([darkness]), was, upon, the, face, of, the, deep])])], s(f)).

str(gen(one('2'(a(i)))), [b([the, earth]), was, b([b([without, form, and, void])])], s(f)).

str(gen(one('2'(a(ii)))), [b([darkness]), was, upon, the, face, of, the, deep], s(f)).

%str(gen(one('2'(b))), ['And', b([the, 'Spirit', of, 'God']), moved, upon, the, face, of, the, waters], s(f)).

%str(gen(one(3)), ['And', b([b([b(['God']), said, let, b([there]), be, light, and, b([there]), was, light])])], s(f)).

%str(gen(one('4'(a))), ['And', b([b(['God']), saw, b([b([the, light, and, that, b([it]), was, good])])], s(f)).

%str(gen(one('4'(b))), [b(['God']), divided, the, light, from, the, darkness], s(f)).

%str(gen(one('5'(a))), ['And', b([b([b(['God']), called, the, light, day, and, the, darkness, he, called, night])])], s(f)).


lex(['007'], iL g iu n(t(s(g))), '007').
lex([a], iL g iu (f iu ((s(f) ci iL n(t(s(g)))) in s(f))/cn(s(g))), [lmd, X, [lmd, Y, [xst, Z, [and, [app, X, Z], [app, Y, Z]]]]]).
lex([admire], 'L'((br((a ie n(a)) - (g ie n(t(s(g))))) bs s(f))/(a ie n(a))), [up, [lmd, X, [lmd, Y, [app, 'Pres', [app, [app, [dn, admire], X], Y]]]]]).
lex(['And'], iL f iu (s(f)/s(f)), [lmd, X, X]).
lex([and], iL f iu(('?'iL s(f)bs ab ab s(f))/iL s(f)), [mapPhin, 0, and]).
lex([and], iL a iu f iu(('?'iL A bs ab ab A)/iL A), [mapPhin, [app, s, 0], and]) :-
      A = br n(a)bs s(f);
      A = s(f)/'!'n(a).
lex([and], iL f iu(('?'iL A bs ab ab A)/iL A), [mapPhin, [app, s, 0], and]) :-
      A = s(f)/a ie n(a).
lex([and], iL w iu a iu b iu f iu ((iL A bs ab ab A)/bg bg iL A), [lmd, X, [lmd, Y, [lmd, Z, [and, [app, Y, Z], [app, X, Z]]]]]) :-
      A = (s(f) ci(((br n(a) bs s(f))rio w(w))/n(b)))licn w(w).
lex([and], iL f iu a iu(('?'iL A bs ab ab A)/iL A), [mapPhin, [app, s, [app, s, 0]], and]) :-
      A = (br n(a) bs s(f))/b ie n(b);
      A = ((br n(a) bs s(f))/b ie n(b))bs(br n(a) bs s(f)).
lex([and], iL f iu a iu((iL A bs ab ab A)/iL A), [lmd, X, [lmd, Y, [lmd, Z, [lmd, W, [and, [app, [app, Y, Z], W], [app, [app, X, Z], W]]]]]]) :-
      A = ((br n(a) bs s(f))/((b ie n(b))+(g ie ((cn(g)/cn(g))iad(cn(g) bs cn(g))))))bs(br n(a) bs s(f)).
lex([and], iL a iu b iu f iu (('?'iL A bs ab ab A)/iL A), [mapPhin, [app, s, [app, s, 0]], and]) :-
      A = ((br n(a)bs s(f))/(c ie n(c)+cp(b)))bs(br n(a)bs s(f));
      A = (br n(a) bs s(f))/pp(b);
      A = ((br n(a)bs s(f))/(c ie n(c)*pp(b)))bs(br n(a) bs s(f));
      A = (br n(a)bs s(f))/'!'n(b).
lex([ate], 'L'((br a ie n(a) bs s(f))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, 'Past', [app, [app, [dn, eat], X], Y]]]]]).
lex([bagels], 'L'(n(t(p(n)))&cn(p(n))), [up, [pair, [app, gen, [dn, bagels]], [dn, bagels]]]).
lex([barn], 'L'cn(s(n)), barn).
lex([be], 'L'((br w([there]) liu s(b))/a ie n(a)), [up, [lmd, X, [app,
[dn, be], X]]]).
lex([before], iL(a iu f iu ((br n(a) bs s(f))bs (br n(a) bs s(f)))/s(f)), [lmd, X, [lmd, Y, [lmd, Z, [app, [app, before, X], [app, Y, Z]]]]]).
lex([beginning], 'L'cn(s(n)), beginning).
lex([believes], 'L'((br g ie n(t(s(g))) bs s(f))/(cp(that) iad 'L's(f))), [up, [lmd, X, [lmd, Y, [app, 'Pres', [app, [app, [dn, believe], X], Y]]]]]).
lex([bill], iL n(t(s(m))), b).
lex([bond], iL n(t(s(m))), b).
lex([book], 'L'cn(s(n)), book).
lex([bought], 'L'((br a ie n(a) bs s(f))/((a ie n(a))* (a ie n(a)))), [up, [lmd, X, [lmd, Y, [app, 'Past', [app, [app, [app, [dn, buy], [fst, X]], [snd, X]], Y]]]]]).
lex([bought], 'L'((br a ie n(a) bs s(f))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, 'Past', [app, [app, [dn, buy], X], Y]]]]]).
lex([by], iL(a iu (((br n(a) bs s(-))bs (br n(a) bs s(-)))/n(a))), [lmd, X, [lmd, Y, [lmd, Z, [and, [eq, Z, X], [app, Y, Z]]]]]).
lex([by], 'L'(n iu (cn(n) bs cn(n))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, [app, [dn, by], X], Y]]]]).
```

```
lex([buys], 'L'((br g ie n(t(s(g))) bs s(f))/((a ie n(a)))* (a ie n(a)))), [up, [lmd, X, [lmd, Y, [app, [app, 'Pres', [app, [app, [dn, buy], [fst, X]], [snd, X]], Y]]]]]).
lex([calls], 'L'((br g ie n(t(s(g))) bs s(f))/a ie ((w([up]))lip n(a))riad(n(a))rip w([up])))), [up, [lmd, X, [lmd, Y, [app, [app, [dn, phone], X], Y]]]]).
lex([catch], 'L'((br a ie n(a) bs s(b))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, [app, [dn, catch], X], Y]]]]).
lex([cezanne], iL n(t(s(m))), c).
lex([cd], 'L'cn(s(n)), cd).
lex([charles], iL n(t(s(m))), c).
lex([clark], iL g iu n(t(s(g))), c).
lex([coffee], 'L'(n(t(s(n)))&cn(s(n))), [up, [pair, [app, gen, [dn, coffee]], [dn, coffee]]]).
% lex([cold], w(cold), 0).
lex([created], 'L'((br a ie n(a) bs s(f))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, 'Past', [app, [app, [dn, create], X], Y]]]]]).
lex([darkness], 'L'(cn(s(n))&n(t(s(n)))), [up, [pair, [dn, darkness], [app, gen, [dn, darkness]]]]).
lex([deep], 'L'cn(s(n)), deep).
lex([did], iL a iu g iu b iu h iu ((((br n(a) bs s(g)) ci (br n(b) bs s(h))))/(c ie br n(c) bs s(f)))bs((br n(a) bs s(g))ci(br n(b) bs s(h)))),
[lmd, X, [lmd, Y, [app, [app, X, Y], Y]]]).
lex([did, too], (((br n(_) bs s(_)) ci (br n(_) bs s(_)))/(br n(_) bs s(_)))bs((br n(_) bs s(_))ci(n(_) bs s(_))),
[lmd, X, [lmd, Y, [app, [app, X, Y], Y]]]).
%lex([doesnt], iL g iu((br n(t(s(g))) bs s(f))/(br n(t(s(g))) bs s(b)))), [lmd, % X, [lmd, Y, [not, [app, X, Y]]]]).
lex([doesnt], iL g iu a iu ((s(g)ci((br n(a)bs s(f))/(br n(a)bs s(b))))in s(g)),
[lmd, X, [not, [app, X, [lmd, Y, [lmd, Z, [app, Y, Z]]]]]]).
lex([dog], 'L'cn(s(n)), dog).
lex([donuts], 'L'(n(t(p(n)))&cn(p(n))), [up, [pair, [app, gen, [dn, donuts]], [dn, donuts]]]).
lex([earth], 'L'cn(s(n)), earth).
lex([eat], 'L'((br a ie n(a) bs s(b))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, [app, [dn, eat], X], Y]]]]).
lex([edinburgh], iL n(t(s(n))), e).
lex([editor], 'L'(g iu cn(s(g))/pp(of)), editor).
lex([every], iL g iu (f iu ((s(f) ci n(t(s(g)))) in s(f))/cn(s(g))), [lmd, X, [lmd, Y, [all, Z, [imply, [app, X, Z], [app, Y, Z]]]]]).
lex([everyone], 'L' f iu ((s(f) ci g iu n(t(g))) in s(f)), [up, [lmd, X, [all, Y, [imply, [app, [dn, person], Y], [app, X, Y]]]]]).
lex([face], 'L'cn(s(n)), face).
lex([fell], 'L'(a ie br n(a)bs s(f)), [up, [lmd, X, [app, 'Past', [app, [dn, fall], X]]]]).
lex([filed], 'L'((br g ie n(t(s(g))) bs s(f))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, 'Past', [app, [app, [dn, file], X], Y]]]]]).
lex([finds], 'L'((br g ie n(t(s(g))) bs s(f))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, 'Pres', [app, [app, [dn, find], X], Y]]]]]).
lex([fish], 'L'cn(s(n)), fish).
lex([for], iL(pp(for)/a ie n(a)), [lmd, X, X]).
lex([form], 'L'(cn(s(n))&n(t(s(n)))), [up, [pair, [dn, form], [app, gen, [dn, form]]]]).
lex([fortunately], 'L'f iu (sp s(f))in s(f)), fortunately).
lex([friends], 'L'(cn(p)/pp(of)), friends).
lex([from], 'L'((a iu f iu ((br n(a)bs s(f))bs(br n(a)bs s(f)))&(n iu (cn(n)bs cn(n))))/(b ie n(b))), [up, [lmd, X, [pair, [app, [dn, fromadv], X], [app, [dn, fromadn], X]]]]).
%lex([is], iL((g ie n(t(s(g)))bs s(f))/(g ie (cn(g) bs cn(g)))), [lmd, X, [lmd, Y, [app, [app, X, [lmd, Z, [eq, Z, Y]]], Y]]]).
lex([gave], 'L'((br a ie n(a) bs s(f))/(b ie n(b)*pp(to))), [up, [lmd, X, [lmd, Y, [app, 'Past', [app, [app, [dn, give], [snd, X]], [fst, X]], Y]]]]).
lex([gave], 'L'((br g ie n(t(s(g))) bs s(f))/(a ie n(a) rip w([the, cold, shoulder]))), [up, [lmd, X, [lmd, Y, [app, 'Past', [app, [app, [dn, shun], X], Y]]]]]).
lex([gave], 'L'(((br (a ie n(a)) bs s(f))/(a ie n(a)))/(a ie n(a))), [up, [lmd, X, [lmd, Y, [lmd, Z, [app, 'Past', [app, [app, [app, [dn, give], X], Y], Z]]]]]]).
%lex([gave, I, the, cold, shoulder], (n(_) bs s(f)) ci n(_), shun).
lex([girl], 'L'cn(s(f)), girl).
lex([gives], 'L'((br g ie n(t(s(g))) bs s(f))/(a ie n(a) rip w([the, cold, shoulder]))), [up, [lmd, X, [lmd, Y, [app, 'Pres', [app, [app, [dn, shun], X], Y]]]]]).
lex(['God'], iL n(t(s(m))), 'God').
lex([good], 'L'n iu (cn(n)/cn(n)), good).
lex([has], 'L'((br g ie n(t(s(g))) bs s(f))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, 'Pres' , [app, [app, [dn, have], X], Y]]]]]).
lex([he], iL ab g iu ((iL s(g) lca iL n(t(s(m)))/(br n(t(s(m))) bs s(g))), [lmd,X,X]).
lex([heaven], 'L'cn(s(n)), heaven).
lex([her], iL g iu a iu (((br n(a)bs s(g)) ci iL n(t(s(f)))) in (iL (br n(a)bs s(g)) lca iL n(t(s(f))))), [lmd,X,X]).
lex([himself], iL f iu(((br n(t(s(m))) bs s(f))ci n(t(s(m))))in((br n(t(s(m))) bs s(f)))), [lmd, X, [lmd, Y, [app, [app, X, Y], Y]]]).
lex([horse], 'L'cn(s(n)), horse).
lex([in], 'L'(a iu f iu((br n(a) bs s(f)) bs (br n(a) bs s(f)))/a ie n(a)), [up, [lmd, X, [lmd, Y, [lmd, Z, [app, [app, [dn, in], X], [app, Y, Z]]]]]]).
lex([in], 'L' (f iu (s(f) nd s(f))/ a ie n(a)), in).
lex([is], iL((br g ie n(t(s(g)))bs s(f))/((a ie n(a))+(g ie ((cn(g)/cn(g))iad(cn(g) bs cn(g))
)-1))), [lmd, X, [lmd, Y, [app, 'Pres', [case, X, Z, [eq, Y, Z], W, [app, [app, W, [lmd, U, [eq, U, Y]]], Y]]]]]).
lex([it], iL w([it]), 0).
%lex([it], iL f iu ((s(f) ci iL n(t(s(n)))) in (iL s(f) lca iL n(t(s(n)))))), [lmd,X,X]).
lex([it], iL f iu a iu (((br n(a)bs s(f)) ci iL n(t(s(n)))) in (iL (br n(a)bs s(f)) lca iL n(t(s(n)))))), [lmd,X,X]).
lex([it], iL ab f iu ((iL s(f) lca iL n(t(s(n)))/(br n(t(s(n))) bs s(f))),
[lmd,X,X]).
lex([jogs], 'L'(br g ie n(t(s(g)))bs s(f)), [up, [lmd, X, [app, 'Pres' , [app, [dn, jog], X]]]]).
lex([john], iL n(t(s(m))), j).
lex([laughs], 'L'(br g ie n(t(s(g))) bs s(f)), [up, [lmd, X, [app, 'Pres', [app, [dn, laugh], X]]]]).
lex([left], 'L'(br g ie n(t(s(g)))bs s(f)), [up, [lmd, X, [app, 'Pres' , [app, [dn, leave], X]]]]).
lex([let], 'L'(s(im)/s(b)), let).
%lex([light], 'L'n(t(s(n))), light).
lex([light], 'L'(cn(s(n))&n(t(s(n)))), [up, [pair, [dn, light], [app, gen, [dn, light]]]]).
lex([likes], 'L'((br g ie n(t(s(g)))bs s(f))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, 'Pres' , [app, [app, [dn, like], X], Y]]]]]).
lex([logic], 'L'(n(t(s(n)))&cn(s(n))), [up, [pair, [app, gen, [dn, logic]], [dn, logic]]]).
```

```
lex([london], iL n(t(s(n))), l).
% lex([loved], 'L'(a iu (b iu (((g ie (cn(g)bs cn(g))/(n(a)bs(n(b)bs s(-))))*(n(a)bs%(n(b)bs s(-)))))),
%[up, [pair, [lmd, X, [lmd, Y, [lmd, Z, [and, [app, Y, Z], [xst, W, [app, [app, X, %Z], W]]]]], [dn, love]]]).
%lex([loved], 'L'(a iu (b iu (((n(a) bs s(-))ci n(b))dp(((n(a) bs s(-))ci n(b))in(g %iu (cn(g)bs cn(g)))))), (up, [pair, [dn, love], [lmd, X, [lmd, Y, [lmd, Z, [and,
%[app, Y, Z], [xst, W, [app, [app, X, Z], W]]]]]).
lex([loses], 'L'((br g ie n(t(s(g))) bs s(f))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, 'Pres', [app, [app, [dn, lose], X], Y]]]]]).
lex([love], 'L'((br a ie n(a) bs s(b))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, [app, [dn, love], X], Y]]]]).
lex([loved], 'L'(a iu (b iu (((br n(a) bs s(-))ci n(b))dp(((br n(a) bs s(-))ci n(b))in(g iu (cn(g)bs cn(g)))))), (up, [pair, [dn, love], [lmd, X, [lmd, Y, [lmd, Z, [and,
[app, Y, Z], [xst, W, [app, [app, X, Z], W]]]]]]]]).
lex([loves], 'L'((br g ie n(t(s(g))) bs s(f))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, 'Pres', [app, [app, [dn, love], X], Y]]]]]).
lex([man], 'L'cn(s(m)), man).
lex([mary], iL n(t(s(f))), m).
lex([met], 'L'((br a ie n(a)bs s(f))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, 'Past', [app, [app, [dn, meet], X], Y]]]]]).
lex([more], iL h iu g iu f iu ((s(f) ci Q) in(s(f)/bg(cp(than) ci iL Q))), [lmd, X, [lmd, Y, [more_than, [card, [lmd, Z, [app, X,
[lmd,P,[lmd,R,[and,[app,P,Z],[app,R,Z]]]]]], [card, [lmd, Z1, [dn, [app, Y, [lmd, P1, [lmd, Q1, [and, [app, P1, Z1],
[app, Q1, Z1]]]]]]]]]]) :- Q=((s(h) ci n(t(p(g)))) in s(h))/cn(p(g)).
lex([mountain], 'L'cn(s(n)), mountain).
lex([moved], 'L'(br a ie n(a) bs s(f)), [up, [lmd, X, [app, 'Past', [app, [dn, move], X]]]]).
lex([necessarily], iL(s(F)/'L's(F)), 'Nec').
%lex([of], iL(pp(of)/a ie n(a)), [lmd, X, X]).
lex([of], 'L'(((n iu (cn(n) bs cn(n)))/iL (b ie n(b)))&(pp(of)/(a ie n(a)))), [up, [pair, [dn, of], [lmd, X, X]]]).
lex([or], iL f iu (('?'iL s(f)bs ab ab s(f))/iL s(f)), [mapPhin, 0, or]).
lex([or], iL a iu f iu (('?'iL A bs ab ab A)/iL A), [mapPhin, [app, s, 0], or]) :- A = br n(a) bs s(f).%;
%       A = s(f)/(br g ie n(t(s(g))) bs s(f)).
lex([or], iL f iu (('?'iL A bs ab ab A)/iL A), [mapPhin, [app, s, 0], or]) :-
      A = s(f)/(br g ie n(t(s(g))) bs s(f)).
lex([or], iL a iu f iu(('?'iL A bs ab ab A)/iL A), [mapPhin, [app, s, [app, s, [app, s, 0]]], or]) :-
      A = (((br n(a)) bs s(f))/(b ie n(b)))/(b ie n(b)).
lex([painting], 'L'(cn(s(n))/pp(of)), [up, [lmd, X, [app, [app, [dn, of], X], [dn, painting]]]]).
lex([paper], 'L'cn(s(n)), paper).
lex([park], 'L'cn(s(n)), park).
lex([past], 'L'a iu f iu(((br n(a)bs s(f))bs (br n(a)bs s(f)))/(b ie n(b))), [up, [lmd, X, [lmd, Y, [lmd, Z, [app, [app, [dn, past], X], [app, Y, Z]]]]]]).
lex([perseverance], 'L'(n(t(s(n)))&cn(s(n))), [up, [pair, [app, gen, [dn, perseverance]], [dn, perseverance]]]).
lex([peter], iL n(t(s(m))), p).
lex([phonetics], 'L'(n(t(s(n)))&cn(s(n))), [up, [pair, [app, gen, [dn, phonetics]], [dn, phonetics]]]).
lex([praises], 'L'((br (g ie n(t(s(g)))) bs s(f))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, 'Pres', [app, [app, [dn, praise], X], Y]]]]]).
lex([raced], 'L'(br(a ie n(a))bs s(f)), [up, [lmd, X, [app, 'Past', [app, [dn, race], X]]]]).
%lex([raced], ((cn(G)bs cn(G))/(n(A)bs(n(B)bs s(-))))*(n(A)bs(n(B)bs s(-))),
%[pair, [lmd, X, [lmd, Y, [lmd, Z, [and, [app, Y, Z], [xst, W, [app, [app, X, Z], %W]]]]], race2]).
lex([raced], 'L'(a iu (b iu (((br n(a) bs s(-))ci n(b))dp(((br n(a) bs s(-))ci n(b))in(g iu (cn(g)bs cn(g)))))), (up, [pair, [dn, race2], [lmd, X, [lmd, Y, [lmd, Z, [and,
[app, Y, Z], [xst, W, [app, [app, X, Z], W]]]]]]]]).
lex([rains], 'L'(br w([it]) liu s(f)), [up, [app, 'Pres', [dn, itrains]]]).
lex([reading], 'L'((br a ie n(a) bs s(psp))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, [app, [dn, read], X], Y]]]]).
lex([robin], iL g iu n(t(s(g))), r).
lex([said], 'L'((br a ie n(a) bs s(f))/s(im)), [up, [lmd, X, [lmd, Y, [app, 'Past',
[app, [app, [dn, say], X], Y]]]]]).
%lex([saw], 'L'((br g ie n(t(s(g))) bs s(f))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, 'Past', [app, [app, [dn, see], X], Y]]]]]).
lex([saw], 'L'((br a ie n(a) bs s(f))/(a ie n(a)+cp(that))), [up, [lmd, X, [lmd, Y,
[app, 'Past', [app, [case, X, X1, [app, [dn, seee], X1], X2, [app, [dn, seet], X2]], Y]]]]]).
lex([seeks], 'L'((br g ie n(t(s(g))) bs s(f))/'L'a iu f iu (((n(a) bs s(f))/b ie n(b)) bs (n(a) bs s(f)))), [up, [lmd, X, [lmd, Y, [app, [app, [dn, tries], [up, [app, [app, [dn, X], [dn, find]], Y]]], Y]]]]).
lex([sees], 'L'((br g ie n(t(s(g))) bs s(f))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, 'Pres', [app, [app, [dn, see], X], Y]]]]]).
lex([sent], 'L'((br a ie n(a) bs s(f))/(b ie n(b)*pp(to))), [up, [lmd, X, [lmd, Y, [app, 'Past', [app, [app, [app, [dn, sent], [snd, X]], [fst, X]], Y]]]]]).
lex([sent], 'L'(((br (a ie n(a)) bs s(f))/(a ie n(a)))/(a ie n(a))), [up, [lmd, X, [lmd, Y, [lmd, Z, [app, 'Past', [app, [app, [app, [dn, send], X], Y], Z]]]]]]).
lex([she], iL ab g iu ((iL s(g) lca iL n(t(s(f))))/(br n(t(s(f))) bs s(g))), [lmd,X,X]).
% lex([shoulder], w(shoulder), 0).
lex([sings], 'L'(br g ie n(t(s(g)))bs s(f)), [up, [lmd, X, [app, 'Pres', [app, [dn, sing], X]]]]).
lex([slept], 'L'(br g ie n(t(s(g)))bs s(f)), [up, [lmd, X, [app, 'Past', [app, [dn, sleep], X]]]]).
lex([slowly], 'L'a iu f iu('L'(br n(a) bs s(f)) bs (br 'L'n(a) bs s(f))), [up, [lmd, X, [lmd, Y, [app, [dn, slowly], [up, [app, [dn, X], [dn, Y]]]]]]]).
lex([sneezed], 'L'(br g ie n(t(s(g)))bs s(f)), [up, [lmd, X, [app, 'Past', [app, [dn, sneeze], X]]]]).
lex([sold], 'L'((br a ie n(a) bs s(f))/(b ie n(b)*pp(for))), [up, [lmd, X, [lmd, Y, [app, 'Past', [app, [app, [app, [dn, sell], [snd, X]], [fst, X]], Y]]]]).
lex([someone], 'L' f iu ((s(f) ci iL g iu n(t(g))) in s(f)), [up, [lmd, X, [xst, Y, [and, [app, [dn, person], Y], [app, X, Y]]]]].
lex(['Spirit'], 'L'cn(s(m)), 'Spirit').
lex([studies], 'L'((br g ie n(t(s(g))) bs s(f))/a ie n(a)), [up, [lmd, X, [lmd, Y, [app, 'Pres', [app, [app, [dn, study], X], Y]]]]]).
lex([such, that], iL n iu ((cn(n) bs cn(n))/(s(f) lca iL n(t(n)))), [lmd, X, [lmd, Y, [lmd, Z, [and, [app, Y, Z], [app, X, Z]]]]].
lex([suzy], iL n(t(s(f))), s).
lex([talks], 'L'(br g ie n(t(s(g)))bs s(f)), [up, [lmd, X, [app, 'Pres', [app, [dn, talk], X]]]]).
lex([tall], 'L'(g iu (cn(g)/cn(g))), tall).
lex([teetotal], 'L' n iu (cn(n)/cn(n)), [up, [lmd, X, [lmd, Y, [and, [app, X, Y], [app, [dn, teetotal], Y]]]]]).
lex([tenmilliondollars], 'L'n(t(s(n))), tenmilliondollars).
lex([than], iL(cp(than)/'L's(f)), [lmd, X, X]).
```

```
lex([that], iL(cp(that)/'L's(F)), [lmd, X, X]).
lex([that], iL(n iu (ab ab (cn(n))bs cn(n))/iL((br n(t(g)))iac''iL n(t(n)))bs s(F)))), [lmd, X, [lmd, Y, [lmd, Z, [and, [app, Y, Z], [app, X, Z]]]]]]).
%lex([that], (cn(G)bs cn(G))/(s(F)/n(t(G)))), [lmd, X, [lmd, Y, [lmd, Z, [and, [app, Y, Z], [app, X, Z]]]]]).
lex([the], iL(n iu (n(t(n))/cn(n))), iota).
lex([the, cold, shoulder], iL w([the, cold, shoulder]), 0).
lex([there], iL w([there]), 0).
lex([thinks], 'L'((br g ie n(t(s(g))) bs s(F))/(cp(that)iad'L's(F))), [up, [lmd, X, [lmd, Y, [app, 'Pres', [app, [app, [dn, think], X], Y]]]]]).
lex([to], iL((pp(to)/a ie n(a))iac(n iu ((br n(n) bs s(i))/(br n(n) bs s(b))))), [lmd, X, X]).
lex([today], 'L'a iu f iu ((br n(a)bs s(F))bs(br n(a)bs s(F))), [up, [lmd, X, [lmd, Y, [app, [dn, today], [app, X, Y]]]]]).
lex([tries], 'L'((br g ie n(t(s(g))) bs s(F))/'L'(br g ie n(t(s(g))) bs s(i))), [up, [lmd, X, [lmd, Y, [app, [app, [dn, tries], [up, [app, [dn, X], Y]]], Y]]]]).
lex([unicorn], 'L'cn(s(n)), unicorn).
lex([up], iL w([up]), 0).
lex([upon], 'L'((b iu f iu ((br n(b) bs s(F))bs(br n(b) bs s(F)))&g iu (cn(g) bs cn(g)))/a ie n(a)), [up, [lmd, X, [pair, [app, [dn, uponadv], X], [app, [dn, uponadn], X]]]]).
lex([void], 'L'g iu (cn(g)/cn(g)), void).
lex([walk], 'L'(br ((a ie n(a)) - (g ie n(t(s(g))))) bs s(F)), [up, [lmd, X, [app, 'Pres', [app, [dn, walk], X]]]]).
lex([walk], 'L'(br a ie n(a)bs s(b)), [up, [lmd, X, [app, [dn, walk], X]]]).
lex([walks], 'L'(br g ie n(t(s(g)))bs s(F)), [up, [lmd, X, [app, 'Pres', [app, [dn, walk], X]]]]).
lex([was], iL((br g ie n(t(s(g)))bs s(F))/((a ie n(a))+(g ie ((cn(g)/cn(g))iad(cn(g) bs cn(g)))-1))), [lmd, X, [lmd, Y, [app, 'Past', [case, X, Z, [eq, Y, Z], W, [app, [app, W, [lmd, U, [eq, U, Y]]], Y]]]]]).
lex([was], 'L'((br w([there]) )iu s(F))/a ie n(a)), [up, [lmd, X, [app, 'Past', [app, [dn, be], X]]]]).
lex([waters], 'L'cn(p(n)), waters).
lex([which], iL(n iu m iu ((a(t(a))ci n(t(m)))in (ab ab (cn(m)bs cn(m))/iL((br n(t(n))iac''iL n(t(n)))bs s(F))))), [lmd, W, [lmd, X, [lmd, Y, [lmd, Z, [and, [app, Y, Z], [app, X, [app, W, Z]]]]]]]).
lex([who], iL (h iu n iu (ab ab (n(t(n)))bs((s(b) ci n(t(n)))in s(b)))/iL((br n(t(n)))iac''iL n(t(n)))bs s(F)))), [lmd, X, [lmd, Y, [lmd, Z, [and, [app, X, Y], [app, Z, Y]]]]]).
lex([will], iL a iu((br n(a)bs s(F))/(br n(a)bs s(b))), [lmd, X, [lmd, Y, [app, 'Fut', [app, X, Y]]]]).
lex([without], 'L'(g iu (cn(g) bs cn(g))/ a ie n(a)), [up, [lmd, X, [lmd, Y, [lmd, Z, [and, [app, Y, Z], [not, [app, [app, [dn, with], X], Z]]]]]]]).
lex([without], iL a iu f iu(ab(br n(a)bs s(F))bs(br n(a)bs s(F)))/(br n(a) bs s(psp))), [lmd, X, [lmd, Y, [lmd, Z, [and, [app, Y, Z], [not, [app, X, Z]]]]]]]).
lex([woman], 'L'cn(s(F)), woman).
lex([yesterday], 'L'a iu f iu ((br n(a)bs s(F))bs(br n(a)bs s(F))), [up, [lmd, X, [lmd, Y, [app, [dn, yesterday], [app, X, Y]]]]]).

onecoord(A) :- member(A, [%,
%s(F)/(br (g ue 'L'n(t(s(g)))) bs s(F)),
%s(_)/n(_),
%(s(F)/(g ue n(g)))bs s(F)
]).
```

Execution of t(*N*) creates a file t.tex containing LaTeX source for the translations and proofs computed. Consider continuing for example:

```
?- t(1).
```

For this, t.tex is:

```
\vspace{0.15in}
(1) $[{\bf john}]{\rightarrow}{\bf walks}: Sf$

\vspace{0.15in}
$[{\blacksquare}Nt(s(m)): {\it j}], {\square}({{\langle\rangle}{\exists}gNt(s(g))\backslash Sf): \mbox{\`I}}\lambda A({{\it Pres}\ (\mbox{\v{}}{\it walk}\ {\it A}))\ \Rightarrow\ Sf$

\vspace{0.15in}

\prooftree
\prooftree
\prooftree
\prooftree
\prooftree
\prooftree
\justifies
\mbox{\fbox{$Nt(s(m))$}}\ \Rightarrow\ Nt(s(m))
\endprooftree
\justifies
\mbox{\fbox{${\blacksquare}Nt(s(m))$}}\ \Rightarrow\ Nt(s(m))
\using {\blacksquare}L
\endprooftree
\justifies
{\blacksquare}Nt(s(m))\ \Rightarrow\ \fbox{${\exists}gNt(s(g))$}
\using {\exists}R
\endprooftree
\justifies
[{\blacksquare}Nt(s(m))]\ \Rightarrow\ \fbox{${\langle\rangle}{\exists}gNt(s(g))$}
\using {\langle\rangle}R
\endprooftree
\prooftree
\justifies
\mbox{\fbox{$Sf$}}\ \Rightarrow\ Sf
\endprooftree
\justifies
[{\blacksquare}Nt(s(m))], \mbox{\fbox{${\langle\rangle}{\exists}gNt(s(g))\backslash Sf$}}\ \Rightarrow\ Sf
\using {\backslash}L
\endprooftree
\justifies
[[{\blacksquare}Nt(s(m))], \mbox{\fbox{${\square}({\langle\rangle}{\exists}gNt(s(g))\backslash Sf)$}}\ \Rightarrow\ Sf
\using {\Box}L
\endprooftree

\vspace{0.15in}

${\it Pres}\ (\mbox{\v{}}{\it walk}\ {\it j})$
```

Consider consulting f9.1.1.pl and then executing the queries:

?- pplexlatex, t(1).

The lexicon file s.tex is created by the first query and the derivations file t.tex is created by the second query. The file out.tex inputs the lexicon s.tex and derivations t.tex:

```
\documentclass{article}
\usepackage{latexsym}
\usepackage{lingmacros,xypic,prooftree,amssymb,lscape,a4wide}
\usepackage{pxfonts}
\usepackage{stmaryrd}
\usepackage{wasysym}
\setlength{\parindent}{0ex}
\begin{document}

\input{s}

\input{t}

\end{document}
```

Then the result of running LaTeX on out.tex is:

007 : $\blacksquare\forall g N t(s(g))$: *007*
a : $\blacksquare\forall g(\forall f((S f^{\uparrow}\blacksquare N t(s(g)))\backslash S f)/CNs(g)) : \lambda A\lambda B\exists C[(A\ C)\wedge(B\ C)]$
admire : $\square(((\diamond\exists a N a - \exists g N t(s(g)))\backslash S f)/\exists a N a) : \hat{}\lambda A\lambda B(Pres\ (\check{}admire\ A)\ B))$
And : $\blacksquare\forall f(S f/S f) : \lambda A A$
and : $\blacksquare\forall f((?\blacksquare S f)\sqcap^{-1}\sqcap^{-1}S f)/\blacksquare S f) : (\Phi^{n+}\ 0\ and)$
and : $\blacksquare\forall a\forall f((?\blacksquare(\diamond N a\backslash S f)\backslash(\sqcap^{-1}\sqcap^{-1}(\diamond N a\backslash S f)))/\blacksquare(\diamond N a\backslash S f)) : (\Phi^{n+}\ (s\ 0)\ and)$
and : $\blacksquare\forall a\forall f((?\blacksquare(S f/!N a))\backslash(\sqcap^{-1}\sqcap^{-1}(S f/!N a)))/\blacksquare(S f/!N a)) : (\Phi^{n+}\ (s\ 0)\ and)$
and : $\blacksquare\forall f((?\blacksquare(S f/\exists a N a))\backslash(\sqcap^{-1}\sqcap^{-1}(S f/\exists a N a)))/\blacksquare(S f/\exists a N a)) : (\Phi^{n+}\ (s\ 0)\ and)$
and : $\blacksquare\forall w\forall a\forall b\forall f((\blacksquare((S f^{\uparrow}(((\diamond N a\backslash S f)\circ - W a w)/N b))\overset{\wedge}{\_}W a w))\sqcap^{-1}\sqcap^{-1}(S f^{\uparrow}(((\diamond N a\backslash S f)\circ - W a w)/N b))\overset{\wedge}{\_}W w))/\hat{}\blacksquare((S f^{\uparrow}(((\diamond N a\backslash S f)\circ - W a w)/N b))\overset{\wedge}{\_}W w)) : \lambda A\lambda B\lambda C[(B\ C)\wedge(A\ C)]$
and : $\blacksquare\forall f\forall a((?\blacksquare(((\diamond N a\backslash S f)/\exists b N b)\backslash\sqcap^{-1}\sqcap^{-1}((\diamond N a\backslash S f)/\exists b N b)))\blacksquare((\diamond N a\backslash S f)/\exists b N b)) : (\Phi^{n+}\ (s\ (s\ 0))\ and)$
and : $\blacksquare\forall f\forall a((?\blacksquare(((\diamond N a\backslash S f)/\exists b N b)\backslash(\diamond N a\backslash S f))\backslash\sqcap^{-1}\sqcap^{-1}(((\diamond N a\backslash S f)/\exists b N b)\backslash(\diamond N a\backslash S f)))/\blacksquare(((\diamond N a\backslash S f)/\exists b N b)\backslash(\diamond N a\backslash S f))) : (\Phi^{n+}\ (s\ (s\ 0))\ and)$
and : $\blacksquare\forall f\forall a((?\blacksquare(((\diamond N a\backslash S f)/(\exists b N b\oplus\exists g((C N g/C N g)\sqcup(C N g\backslash C N g))))\backslash(\diamond N a\backslash S f))\backslash\sqcap^{-1}\sqcap^{-1}(((\diamond N a\backslash S f)/(\exists b N b\oplus\exists g((C N g/C N g)\sqcup(C N g\backslash C N g))))\backslash(\diamond N a\backslash S f)))/\blacksquare(((\diamond N a\backslash S f)/(\exists b N b\oplus\exists g((C N g/C N g)\sqcup(C N g\backslash C N g))))\backslash(\diamond N a\backslash S f))) : (\Phi^{n+}\ (s\ (s\ 0))\ and)$
$\lambda A\lambda B\lambda C\lambda D[((B\ C)\ D)\wedge((A\ C)\ D)]$
and : $\blacksquare\forall a\forall b\forall f((?\blacksquare(((\diamond N a\backslash S f)/(\exists c N c\oplus C P b)))\backslash(\sqcap^{-1}\sqcap^{-1}(((\diamond N a\backslash S f)/(\exists c N c\oplus C P b)))\backslash(\diamond N a\backslash S f)))/\blacksquare(((\diamond N a\backslash S f)/(\exists c N c\oplus C P b)))\backslash(\diamond N a\backslash S f))) : (\Phi^{n+}\ (s\ (s\ 0))\ and)$
and : $\blacksquare\forall a\forall b\forall f((?\blacksquare(((\diamond N a\backslash S f)/P P b)\backslash(\sqcap^{-1}\sqcap^{-1}((\diamond N a\backslash S f)/P P b)))\backslash(\sqcap^{-1}\sqcap^{-1}(((\diamond N a\backslash S f)/(\exists c N c\bullet P P b)))\backslash(\diamond N a\backslash S f)))/\blacksquare(((\diamond N a\backslash S f)/(\exists c N c\bullet P P b)))\backslash(\diamond N a\backslash S f))) : (\Phi^{n+}\ (s\ (s\ 0))\ and)$
and : $\blacksquare\forall a\forall b\forall f((?\blacksquare(((\diamond N a\backslash S f)/!N b)\backslash(\sqcap^{-1}\sqcap^{-1}(((\diamond N a\backslash S f)/!N b)))\blacksquare(((\diamond N a\backslash S f)/!N b)) : (\Phi^{n+}\ (s\ (s\ 0))\ and)$
ate : $\square(((\diamond\exists a N a\backslash S f)/\exists a N a) : \hat{}\lambda A\lambda B(Past\ (\check{}eat\ A)\ B))$
bagels : $\square(N t(p(n))\&C N p(n)) : \hat{}(gen\ \check{}bagels),\ \check{}bagels)$
barn : $\square C N s(n) : barn$
be : $\square(((\diamond W[there]\multimap S b)/\exists a N a) : \hat{}\lambda A(\check{}be\ A)$
before : $\blacksquare(\forall a\forall f(((\diamond N a\backslash S f)\backslash(\diamond N a\backslash S f)))/S f) : \lambda A\lambda B\lambda C((before\ A)\ (B\ C))$
beginning : $\square C N s(n) : beginning$
believes : $\square(((\diamond\exists g N t(s(g)))\backslash S f)/(CP\hat{}that\sqcup\square S f)) : \lambda A\lambda B(Pres\ (\check{}believe\ A)\ B))$
bill : $\blacksquare N t(s(m)) : b$
bond : $\blacksquare N t(s(m)) : b$
book : $\square C N s(n) : book$
bought : $\square(((\diamond\exists a N a\backslash S f)/(\exists a N a\bullet\exists a N a)) : \hat{}\lambda A\lambda B(Past\ (((\check{}buy\ \pi_1 A)\ \pi_2 A)\ B))$
bought : $\square(((\diamond\exists a N a\backslash S f)/\exists a N a) : \hat{}\lambda A\lambda B(Past\ (\check{}buy\ A)\ B))$
by : $\blacksquare\forall a(((\diamond N a\backslash S -))\backslash((\diamond N a\backslash S -))/(N a)) : \lambda A\lambda B\lambda C[[C = A]\wedge(B\ C)]$
by : $\square(\forall n(C N n\backslash C N n)/\exists a N a) : \hat{}\lambda A\lambda B(\check{}by\ A\ B)$
buys : $\square(((\diamond\exists g N t(s(g)))\backslash S f)/(\exists a N a\bullet\exists a N a)) : \hat{}\lambda A\lambda B(Pres\ (((\check{}buy\ \pi_1 A)\ \pi_2 A)\ B))$
calls : $\square(((\diamond\exists g N t(s(g)))\backslash S f)/\exists a((W[up]\bullet(\diamond N a)\sqcup(N a\bullet W[up])))) : \hat{}\lambda A\lambda B(\check{}phone\ A)\ B)$
catch : $\square(((\diamond\exists a N a\backslash S b)/\exists a N a) : \hat{}\lambda A\lambda B(\check{}catch\ A)\ B)$
cezanne : $\blacksquare N t(s(m)) : c$
cd : $\square C N s(n) : cd$
charles : $\blacksquare N t(s(m)) : c$
clark : $\blacksquare\forall g N t(s(g)) : c$
coffee : $\square(N t(s(n))\&C N s(n)) : \hat{}((gen\ \check{}coffee),\ \check{}coffee)$
created : $\square(((\diamond\exists a N a\backslash S f)/\exists a N a) : \hat{}\lambda A\lambda B(Past\ (\check{}create\ A)\ B))$
darkness : $\square(CN s(n)\&N t(s(n))) : \hat{}(\check{}darkness,\ (gen\ \check{}darkness))$
deep : $\square C N s(n) : deep$
did : $\blacksquare\forall a\forall b\forall h((((\diamond N a\backslash S g)\uparrow(\diamond N b\backslash S h))/(\exists c(\diamond N c\backslash S f))\backslash(((\diamond N a\backslash S g)\uparrow(\diamond N b\backslash S h)))) : \lambda A\lambda B((A\ B)\ B)$
did+too : $(((\diamond N A\backslash S B)\uparrow(\diamond N C\backslash S D))/((\diamond N E\backslash S F))\backslash(((\diamond N G\backslash S H)\uparrow(N I\backslash S J))) : \lambda K\lambda L((K\ L)\ L)$
doesnt : $\blacksquare\forall g\forall a((S g^{\uparrow}((\diamond N a\backslash S f)/((\diamond N a\backslash S b))))\backslash S g) : \lambda A\neg(A\ \lambda B\lambda C(B\ C))$
dog : $\square C N s(n) : dog$
donuts : $\square(N t(p(n))\&C N p(n)) : \hat{}((gen\ \check{}donuts),\ \check{}donuts)$
earth : $\square C N s(n) : earth$
eat : $\square(((\diamond\exists a N a\backslash S b)/\exists a N a) : \hat{}\lambda A\lambda B(\check{}eat\ A)\ B)$
edinburgh : $\blacksquare N t(s(n)) : e$
editor : $\square(\forall g C N s(g)/P P o f) : editor$
every : $\blacksquare\forall g(\forall f((S f^{\uparrow}N t(s(g)))\backslash S f)/C N s(g)) : \lambda A\lambda B\forall C[(A\ C)\rightarrow(B\ C)]$

everyone : $\Box\forall f((Sf\uparrow\forall gNt(g))\downarrow Sf) : \hat{}\lambda A\forall B[(\check{}person\ B) \rightarrow (A\ B)]$
face : $\Box CNs(n) : face$
fell : $\Box(\exists a\langle\rangle Na\backslash Sf) : \hat{}\lambda A(Past\ (\check{}fall\ A))$
filed : $\Box((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Past\ ((\check{}file\ A)\ B))$
finds : $\Box((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}find\ A)\ B))$
fish : $\Box CNs(n) : fish$
for : $\blacksquare(PPfor/\exists aNa) : \lambda AA$
form : $\Box(CNs(n)\&Nt(s(n))) : \hat{}(\check{}form, (gen\ \check{}form))$
fortunately : $\Box\forall f(\check{}Sf\downarrow Sf) : fortunately$
friends : $\Box(CNp/PPof) : friends$
from : $\Box((\forall a\forall f((\langle\rangle Na\backslash Sf)\backslash(\langle\rangle Na\backslash Sf))\&\forall n(CNn\backslash CNn))/\exists bNb) : \hat{}\lambda A((\check{}fromadv\ A), (\check{}fromadn\ A))$
gave : $\Box((\langle\rangle\exists aNa\backslash Sf)/(\exists bNb\bullet PPto)) : \hat{}\lambda A\lambda B(Past\ (((\check{}give\ \pi_2 A)\ \pi_1 A)\ B))$
gave : $\Box((\langle\rangle\exists gNt(s(g))\backslash Sf)/(\exists aNa\blacktriangleright W[the, cold, shoulder])) : \hat{}\lambda A\lambda B(Past\ ((\check{}shun\ A)\ B))$
gave : $\Box(((\langle\rangle\exists aNa\backslash Sf)/\exists aNa)/\exists aNa) : \hat{}\lambda A\lambda B\lambda C(Past\ (((\check{}give\ A)\ B)\ C))$
girl : $\Box CNs(f) : girl$
gives : $\Box((\langle\rangle\exists gNt(s(g))\backslash Sf)/(\exists aNa\blacktriangleright W[the, cold, shoulder])) : \hat{}\lambda A\lambda B(Pres\ ((\check{}shun\ A)\ B))$
God : $\blacksquare Nt(s(m)) : God$
good : $\Box\forall n(CNn/CNn) : good$
has : $\Box(((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}have\ A)\ B))$
he : $\blacksquare[]^{-1}\forall g((\blacksquare Sg|\blacksquare Nt(s(m)))/(\langle\rangle Nt(s(m))\backslash Sg)) : \lambda AA$
heaven : $\Box CNs(n) : heaven$
her : $\blacksquare\forall g\forall a(((\langle\rangle Na\backslash Sg)\uparrow\blacksquare Nt(s(f)))\downarrow(\blacksquare(\langle\rangle Na\backslash Sg)|\blacksquare Nt(s(f)))) : \lambda AA$
himself : $\blacksquare\forall f((((\langle\rangle Nt(s(m))\backslash Sf)\uparrow Nt(s(m)))\downarrow(\langle\rangle Nt(s(m))\backslash Sf)) : \lambda A\lambda B((A\ B)\ B)$
horse : $\Box CNs(n) : horse$
in : $\Box(\forall a\forall f((\langle\rangle Na\backslash Sf)\backslash(\langle\rangle Na\backslash Sf))/\exists aNa) : \hat{}\lambda A\lambda B\lambda C((\check{}in\ A)\ (B\ C))$
in : $\Box(\forall f(Sf\div Sf)/\exists aNa) : in$
is : $\blacksquare((\langle\rangle\exists gNt(s(g))\backslash Sf)/(\exists aNa\oplus(\exists g((CNg/CNg)\sqcup(CNg\backslash CNg))-I))) : \lambda A\lambda B(Pres\ (A \rightarrow C.[B = C]; D.((D\ \lambda E[E = B])\ B)))$
it : $\blacksquare W[it] : 0$
it : $\blacksquare\forall f\forall a((((\langle\rangle Na\backslash Sf)\uparrow\blacksquare Nt(s(n)))\downarrow(\blacksquare(\langle\rangle Na\backslash Sf)|\blacksquare Nt(s(n)))) : \lambda AA$
it : $\blacksquare[]^{-1}\forall f((\blacksquare Sf|\blacksquare Nt(s(n)))/(\langle\rangle Nt(s(n))\backslash Sf)) : \lambda AA$
jogs : $\Box(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}jog\ A))$
john : $\blacksquare Nt(s(m)) : j$
laughs : $\Box(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}laugh\ A))$
left : $\Box(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}leave\ A))$
let : $\Box(Sim/Sb) : let$
light : $\Box(CNs(n)\&Nt(s(n))) : \hat{}(\check{}light, (gen\ \check{}light))$
likes : $\Box((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}like\ A)\ B))$
logic : $\Box(Nt(s(n))\&CNs(n)) : \hat{}((gen\ \check{}logic), \check{}logic)$
london : $\blacksquare Nt(s(n)) : l$
loses : $\Box(((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}lose\ A)\ B))$
love : $\Box((\langle\rangle\exists aNa\backslash Sb)/\exists aNa) : \hat{}\lambda A\lambda B((\check{}love\ A)\ B)$
loved : $\Box\forall a\forall b((((\langle\rangle Na\backslash S-)\uparrow Nb)\circledcirc((((\langle\rangle Na\backslash S-)\uparrow Nb)\downarrow\forall g(CNg\backslash CNg))) : \hat{}(\check{}love, \lambda A\lambda B\lambda C[(B\ C) \wedge \exists D((A\ C)\ D)])$
loves : $\Box(((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}love\ A)\ B))$
man : $\Box CNs(m) : man$
mary : $\blacksquare Nt(s(f)) : m$
met : $\Box(((\langle\rangle\exists aNa\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Past\ ((\check{}meet\ A)\ B))$
more : $\blacksquare\forall h\forall g\forall f((Sf\uparrow(((Sh\uparrow Nt(p(g)))\downarrow Sh)/CNp(g)))\downarrow(Sf/\hat{}(CPthan\uparrow\blacksquare(((Sh\uparrow Nt(p(g)))\downarrow Sh)/CNp(g))))) : \lambda A\lambda B[|\lambda C(A\ \lambda D\lambda E[(D\ C) \wedge (E\ C)])| > |\lambda F\check{}(B\ \lambda G\lambda H[(G\ F) \wedge (H\ F)])|]$
mountain : $\Box CNs(n) : mountain$
moved : $\Box(\langle\rangle\exists aNa\backslash Sf) : \hat{}\lambda A(Past\ (\check{}move\ A))$
necessarily : $\blacksquare(SA/\Box SA) : Nec$
of : $\Box((\forall n(CNn\backslash CNn)/\blacksquare\exists bNb)\&(PPof/\exists aNa)) : \hat{}(\check{}of, \lambda AA)$
or : $\blacksquare\forall f((?\blacksquare Sf\backslash[]^{-1}[]^{-1}Sf)/\blacksquare Sf) : (\Phi^{n+}\ 0\ or)$

or : $\blacksquare\forall a\forall f((?\blacksquare(\langle\rangle Na\backslash Sf)\backslash\square^{-1}\square^{-1}(\langle\rangle Na\backslash Sf))/\blacksquare(\langle\rangle Na\backslash Sf)) : (\Phi^{n+}\ (s\ 0))\ or)$
or : $\blacksquare\forall f((?\blacksquare(Sf/(\langle\rangle\exists gNt(s(g))\backslash Sf))\backslash\square^{-1}\square^{-1}(Sf/(\langle\rangle\exists gNt(s(g))\backslash Sf)))/\blacksquare(Sf/(\langle\rangle\exists gNt(s(g))\backslash Sf))) : (\Phi^{n+}\ (s\ 0)\ or)$
or : $\blacksquare\forall a\forall f((?\blacksquare(((\langle\rangle Na\backslash Sf)/\exists bNb)/\exists bNb)\backslash\square^{-1}\square^{-1}(((\langle\rangle Na\backslash Sf)/\exists bNb)/\exists bNb))/\blacksquare(((\langle\rangle Na\backslash Sf)/\exists bNb)/\exists bNb)) : (\Phi^{n+}\ (s\ (s\ 0)))\ or)$
painting : $\square(CNs(n)/PPof) : \hat{}\lambda A(\check{}of\ A)\ \check{}painting)$
paper : $\square CNs(n) : paper$
park : $\square CNs(n) : park$
past : $\square\forall a\forall f(((\langle\rangle Na\backslash Sf)\backslash(\langle\rangle Na\backslash Sf))/\exists bNb) : \hat{}\lambda A\lambda B\lambda C(\check{}past\ A)\ (B\ C))$
perseverance : $\square(Nt(s(n))\&CNs(n)) : \hat{}(\check{}gen\ \check{}perseverance)\ \check{}perseverance)$
peter : $\blacksquare Nt(s(m)) : p$
phonetics : $\square(Nt(s(n))\&CNs(n)) : \hat{}(\check{}gen\ \check{}phonetics)\ \check{}phonetics)$
praises : $\square((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ (\check{}praise\ A)\ B))$
raced : $\square(\langle\rangle\exists aNa\backslash Sf) : \hat{}\lambda A(Past\ (\check{}race\ A))$
raced : $\square\forall a\forall b(((\langle\rangle Na\backslash S-)\lceil Nb\rceil\circledcirc(((\langle\rangle Na\backslash S-)\lceil Nb\rceil\downarrow\forall g(CNg\backslash CNg))) : \hat{}(\check{}race2, \lambda A\lambda B\lambda C[(B\ C)\wedge\exists D((A\ C)\ D)])$
rains : $\square(\langle\rangle W[it]\multimap Sf) : \hat{}(Pres\ \check{}itrains)$
reading : $\square(((\langle\rangle\exists aNa\backslash Spsp)/\exists aNa)\ \hat{}\lambda A\lambda B(\check{}read\ A)\ B)$
robin : $\blacksquare\forall gNt(s(g)) : r$
said : $\square((\langle\rangle\exists aNa\backslash Sf)/Sim) : \hat{}\lambda A\lambda B(Past\ (\check{}say\ A)\ B))$
saw : $\square((\langle\rangle\exists aNa\backslash Sf)/(\exists aNa\oplus CPthat)) : \hat{}\lambda A\lambda B(Past\ ((A\to C(\check{}see\ C);D,(\check{}set\ D))\ B))$
seeks : $\square(((\langle\rangle\exists gNt(s(g))\backslash Sf)/\square\forall a\forall f(((Na\backslash Sf)/\exists bNb)\backslash(Na\backslash Sf)) : \hat{}\lambda A\lambda B((\check{}tries\ \hat{}((\check{}A\ \check{}find)\ B))\ B)$
sees : $\square((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}see\ A)\ B))$
sent : $\square((\langle\rangle\exists aNa\backslash Sf)/(\exists bNb\bullet PPto)) : \hat{}\lambda A\lambda B(Past\ (((\check{}sent\ \pi_2 A)\ \pi_1 A)\ B))$
sent : $\square((\langle\rangle\exists aNa\backslash Sf)/\exists aNa)/\exists aNa) : \hat{}\lambda A\lambda B\lambda C(Past\ (((\check{}send\ A)\ B)\ C))$
she : $\blacksquare[]^{-1}\forall g(\blacksquare Sg\downarrow\blacksquare Nt(s(f)))/(\langle\rangle Nt(s(f)))\downarrow Sg)) : \lambda AA$
sings : $\square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}sing\ A))$
slept : $\square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Past\ (\check{}sleep\ A))$
slowly : $\square\forall a\forall f(\square(\langle\rangle Na\backslash Sf)\backslash(\langle\rangle\square Na\backslash Sf)) : \hat{}\lambda A\lambda B(\check{}slowly\ \hat{}(\check{}A\ \check{}B))$
sneezed : $\square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Past\ (\check{}sneeze\ A))$
sold : $\square((\langle\rangle\exists aNa\backslash Sf)/(\exists bNb\bullet PPfor)) : \hat{}\lambda A\lambda B(Past\ (((\check{}sell\ \pi_2 A)\ \pi_1 A)\ B))$
someone : $\square\forall f((Sf\uparrow\blacksquare\forall gNt(g))\downarrow Sf) : \hat{}\lambda A\exists B[(\check{}person\ B)\wedge(A\ B)]$
Spirit : $\square CNs(m) : Spirit$
studies : $\square((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}study\ A)\ B))$
such+that : $\blacksquare\forall n((CNn\backslash CNn)/(Sf\downarrow\blacksquare Nt(n))) : \lambda A\lambda B\lambda C[(B\ C)\wedge(A\ C)]$
suzy : $\blacksquare Nt(s(f)) : s$
talks : $\square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}talk\ A))$
tall : $\square\forall g(CNg/CNg) : tall$
teetotal : $\square\forall n(CNn/CNn) : \hat{}\lambda A\lambda B[(A\ B)\wedge(\check{}teetotal\ B)]$
tenmilliondollars : $\square Nt(s(n)) : tenmilliondollars$
than : $\blacksquare(CPthan/\square Sf) : \lambda AA$
that : $\blacksquare(CPthat/\square Sf) : \lambda AA$
that : $\blacksquare\forall n(\square^{-1}\square^{-1}(CNn\backslash CNn)/\blacksquare(((\langle\rangle Nt(n)\sqcap\blacksquare Nt(n))\backslash Sf)) : \lambda A\lambda B\lambda C[(B\ C)\wedge(A\ C)]$
the : $\blacksquare\forall n(Nt(n)/CNn) : \iota$
the+cold+shoulder : $\blacksquare W[the, cold, shoulder] : 0$
there : $\blacksquare W[there] : 0$
thinks : $\square(((\langle\rangle\exists gNt(s(g))\backslash Sf)/(CPthat\sqcup\square Sf)) : \hat{}\lambda A\lambda B(Pres\ (\check{}think\ A)\ B))$
to : $\blacksquare((PPto/\exists aNa)\sqcap\forall n(((\langle\rangle Nn\backslash Si)/(\langle\rangle Nn\backslash Sb))) : \lambda AA$
today : $\square\forall a\forall f(((\langle\rangle Na\backslash Sf)\backslash((\langle\rangle Na\backslash Sf)) : \hat{}\lambda A\lambda B(\check{}today\ (A\ B))$
tries : $\square(((\exists gNt(s(g))\backslash Sf)/\square(\langle\rangle\exists gNt(s(g))\backslash Si)) : \hat{}\lambda A\lambda B((\check{}tries\ \hat{}(\check{}A\ B))\ B)$
unicorn : $\square CNs(n) : unicorn$
up : $\blacksquare W[up] : 0$
upon : $\square((\forall b\forall f(((\langle\rangle Nb\backslash Sf)\backslash(\langle\rangle Nb\backslash Sf))\&\forall g(CNg\backslash CNg))/\exists aNa) : \hat{}\lambda A((\check{}uponadv\ A), (\check{}uponadn\ A))$
void : $\square\forall g(CNg/CNg) : void$
walk : $\square(\langle\rangle(\exists aNa-\exists gNt(s(g)))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}walk\ A))$

walk : $\Box(\langle\rangle\exists aNa\backslash Sb) : \hat{}\lambda A(\check{}walk\ A)$
walks : $\Box(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}walk\ A))$
was : $\blacksquare((\langle\rangle\exists gNt(s(g))\backslash Sf)/(\exists aNa\oplus(\exists g((CNg/CNg)\sqcup(CNg\backslash CNg))-I))) : \lambda A\lambda B(Past\ (A \rightarrow C.[B = C]; D.((D\ \lambda E[E = B])\ B)))$
was : $\Box((\langle\rangle W[there]\multimap Sf)/\exists aNa) : \hat{}\lambda A(Past\ (\check{}be\ A))$
waters : $\Box CNp(n) : waters$
which : $\blacksquare\forall n\forall m((Nt(n)\uparrow Nt(m))\downarrow([]^{-1}[]^{-1}(CNm\backslash CNm)/\blacksquare((\langle\rangle Nt(n)\sqcap !\blacksquare Nt(n))\backslash Sf))) : \lambda A\lambda B\lambda C\lambda D[(C\ D) \wedge (B\ (A\ D))]$
who : $\blacksquare\forall h\forall n([]^{-1}[]^{-1}(Nt(n)\backslash((Sh\uparrow Nt(n))\downarrow Sh))/\blacksquare((\langle\rangle Nt(n)\sqcap !\blacksquare Nt(n))\backslash Sf)) : \lambda A\lambda B\lambda C[(A\ B) \wedge (C\ B)]$
will : $\blacksquare\forall a((\langle\rangle Na\backslash Sf)/(\langle\rangle Na\backslash Sb)) : \lambda A\lambda B(Fut\ (A\ B))$
without : $\Box(\forall g(CNg\backslash CNg)/\exists aNa) : \hat{}\lambda A\lambda B\lambda C[(B\ C) \wedge \neg((\check{}with\ A)\ C)]$
without : $\blacksquare\forall a\forall f([]^{-1}((\langle\rangle Na\backslash Sf)\backslash(\langle\rangle Na\backslash Sf))/(\langle\rangle Na\backslash Spsp)) : \lambda A\lambda B\lambda C[(B\ C) \wedge \neg(A\ C)]$
woman : $\Box CNs(f) : woman$
yesterday : $\Box\forall a\forall f((\langle\rangle Na\backslash Sf)\backslash(\langle\rangle Na\backslash Sf)) : \hat{}\lambda A\lambda B(\check{}yesterday\ (A\ B))$

(1) **[john]+walks** : Sf

$[\blacksquare Nt(s(m)) : j], \Box(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}walk\ A)) \Rightarrow Sf$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\boxed{Nt(s(m))} \Rightarrow Nt(s(m))}}{\boxed{\blacksquare Nt(s(m))} \Rightarrow Nt(s(m))}\blacksquare L}{\blacksquare Nt(s(m)) \Rightarrow \boxed{\exists gNt(s(g))}}\exists R}{[\blacksquare Nt(s(m))] \Rightarrow \boxed{\langle\rangle\exists gNt(s(g))}}\langle\rangle R \qquad \dfrac{}{\boxed{Sf} \Rightarrow Sf}}{[\blacksquare Nt(s(m))], \boxed{\langle\rangle\exists gNt(s(g))\backslash Sf} \Rightarrow Sf}\backslash L$$
$$\dfrac{}{[\blacksquare Nt(s(m))], \boxed{\Box(\langle\rangle\exists gNt(s(g))\backslash Sf)} \Rightarrow Sf}\Box L$$

$(Pres\ (\check{}walk\ j))$

Part II

Appendix A. Corpus of Examples: text output

Consider consulting f9.1.1.pl and then executing the queries:

?- pplexlatex, t(_).

This generates the following text in the Prolog console.

```
GNU Prolog - version 1.3.0 of Nov 21 2006.
Copyright (C) 1999-2006 Daniel Diaz

compiling /Users/glynmorrill/Glyn/CatLog/f9.1.1.pl for byte code...
/Users/glynmorrill/Glyn/CatLog/f9.1.1.pl compiled, 5912 lines read - 1111535 bytes written, 549 ms


(dwp((7-7))) [john]+walks Sf


[iLNt(s(m)): j], L(<>EgNt(s(g))\Sf): ^LA(Pres (vwalk A)) => Sf


[iLNt(s(m))], L(<>EgNt(s(g))\Sf) => Sf [LL]
    [iLNt(s(m))], <>EgNt(s(g))\Sf => Sf [\L]
        [iLNt(s(m))] => <>EgNt(s(g)) [<>R]
            iLNt(s(m)) => EgNt(s(g)) [ER]
                iLNt(s(m)) => Nt(s(m)) [iLL]
                    Nt(s(m)) => Nt(s(m))
        Sf => Sf

(Pres (vwalk j))


(dwp((7-16))) [every+man]+talks Sf


[iLAg(Af((SfciNt(s(g)))inSf)/CNs(g)): LALBAC[(A C) -> (B C)], LCNs(m): man], L(<>EgNt(s(g))\Sf): ^LD(Pres (vtalk D)) => Sf


[iLAg(Af((SfciNt(s(g)))inSf)/CNs(g)), LCNs(m)], L(<>EgNt(s(g))\Sf) => Sf [iLL]
    [Ag(Af((SfciNt(s(g)))inSf)/CNs(g)), LCNs(m)], L(<>EgNt(s(g))\Sf) => Sf [AL]
        [Af((SfciNt(s(m)))inSf)/CNs(m), LCNs(m)], L(<>EgNt(s(g))\Sf) => Sf [/L]
            LCNs(m) => CNs(m) [LL]
                CNs(m) => CNs(m)
            [Af((SfciNt(s(m)))inSf)], L(<>EgNt(s(g))\Sf) => Sf [AL]
                [(SfciNt(s(m)))inSf], L(<>EgNt(s(g))\Sf) => Sf [inL]
                    [1], L(<>EgNt(s(g))\Sf) => SfciNt(s(m)) [ciR]
                        [Nt(s(m))], L(<>EgNt(s(g))\Sf) => Sf [LL]
                            [Nt(s(m))], <>EgNt(s(g))\Sf => Sf [\L]
                                [Nt(s(m))] => <>EgNt(s(g)) [<>R]
                                    Nt(s(m)) => EgNt(s(g)) [ER]
                                        Nt(s(m)) => Nt(s(m))
                                Sf => Sf
                    Sf => Sf

AC[(vman C) -> (Pres (vtalk C))]


(dwp((7-19))) [the+fish]+walks Sf


[iLAn(Nt(n)/CNn): iota, LCNs(n): fish], L(<>EgNt(s(g))\Sf): ^LA(Pres (vwalk A)) => Sf


[iLAn(Nt(n)/CNn), LCNs(n)], L(<>EgNt(s(g))\Sf) => Sf [LL]
    [iLAn(Nt(n)/CNn), LCNs(n)], <>EgNt(s(g))\Sf => Sf [\L]
        [iLAn(Nt(n)/CNn), LCNs(n)] => <>EgNt(s(g)) [<>R]
            iLAn(Nt(n)/CNn), LCNs(n) => EgNt(s(g)) [ER]
                iLAn(Nt(n)/CNn), LCNs(n) => Nt(s(n)) [iLL]
                    An(Nt(n)/CNn), LCNs(n) => Nt(s(n)) [AL]
                        Nt(s(n))/CNs(n), LCNs(n) => Nt(s(n)) [/L]
                            LCNs(n) => CNs(n) [LL]
                                CNs(n) => CNs(n)
                            Nt(s(n)) => Nt(s(n))
        Sf => Sf

(Pres (vwalk (iota vfish)))
```

```
(dwp((7-32))) [every+man]+[[walks+or+talks]] Sf


[iLAg(Af((SfciNt(s(g)))inSf)/CNs(g)): LALBAC[(A C) -> (B C)], LCNs(m): man], [[L(<>EgNt(s(g))\Sf): ^LD(Pres (vwalk D)), iLAf((?iLSf\[]-1[]-1Sf)/iLSf): [mapPhin,0,or], L(<>EgNt(s(g))\Sf): ^LE(Pres (vtalk E))]] => Sf


[iLAg(Af((SfciNt(s(g)))inSf)/CNs(g)): LALBAC[(A C) -> (B C)], LCNs(m): man], [[L(<>EgNt(s(g))\Sf): ^LD(Pres (vwalk D)), iLAaAf((?iL(<>Na\Sf)\[]-1[]-1(<>Na\Sf))/iL(<>Na\Sf)): [mapPhin,[app,s,0],or], L(<>EgNt(s(g))\Sf): ^LE(Pres (vtalk E))]] => Sf


[iLAg(Af((SfciNt(s(g)))inSf)/CNs(g)), LCNs(m)], [[L(<>EgNt(s(g))\Sf), iLAaAf((?iL(<>Na\Sf)\[]-1[]-1(<>Na\Sf))/iL(<>Na\Sf)), L(<>EgNt(s(g))\Sf)]] => Sf [iLL]
    [Ag(Af((SfciNt(s(g)))inSf)/CNs(g)), LCNs(m)], [[L(<>EgNt(s(g))\Sf), iLAaAf((?iL(<>Na\Sf)\[]-1[]-1(<>Na\Sf))/iL(<>Na\Sf)), L(<>EgNt(s(g))\Sf)]] => Sf [AL]
        [Af((SfciNt(s(m)))inSf)/CNs(m), LCNs(m)], [[L(<>EgNt(s(g))\Sf), iLAaAf((?iL(<>Na\Sf)\[]-1[]-1(<>Na\Sf))/iL(<>Na\Sf)), L(<>EgNt(s(g))\Sf)]] => Sf [/L]
            LCNs(m) => CNs(m) [LL]
                CNs(m) => CNs(m)
            [Af((SfciNt(s(m)))inSf)], [[L(<>EgNt(s(g))\Sf), iLAaAf((?iL(<>Na\Sf)\[]-1[]-1(<>Na\Sf))/iL(<>Na\Sf)), L(<>EgNt(s(g))\Sf)]] => Sf [AL]
                [(SfciNt(s(m)))inSf], [[L(<>EgNt(s(g))\Sf), iLAaAf((?iL(<>Na\Sf)\[]-1[]-1(<>Na\Sf))/iL(<>Na\Sf)), L(<>EgNt(s(g))\Sf)]] => Sf [inL]
                    [1], [[L(<>EgNt(s(g))\Sf), iLAaAf((?iL(<>Na\Sf)\[]-1[]-1(<>Na\Sf))/iL(<>Na\Sf)), L(<>EgNt(s(g))\Sf)]] => SfciNt(s(m)) [ciR]
                        [Nt(s(m))], [[L(<>EgNt(s(g))\Sf), iLAaAf((?iL(<>Na\Sf)\[]-1[]-1(<>Na\Sf))/iL(<>Na\Sf)), L(<>EgNt(s(g))\Sf)]] => Sf [iLL]
                            [Nt(s(m))], [[L(<>EgNt(s(g))\Sf), aAaAf((?iL(<>Na\Sf)\[]-1[]-1(<>Na\Sf))/iL(<>Na\Sf)), L(<>EgNt(s(g))\Sf)]] => Sf [AL]
                                [Nt(s(m))], [[L(<>EgNt(s(g))\Sf), Af((?iL(<>Nt(s(m))\Sf)\[]-1[]-1(<>Nt(s(m))\Sf))/iL(<>Nt(s(m))\Sf)), L(<>EgNt(s(g))\Sf)]] => Sf [AL]
                                    [Nt(s(m))], [[L(<>EgNt(s(g))\Sf), (?iL(<>Nt(s(m))\Sf)\[]-1[]-1(<>Nt(s(m))\Sf))/iL(<>Nt(s(m))\Sf), L(<>EgNt(s(g))\Sf)]] => Sf [/L]
                                        L(<>EgNt(s(g))\Sf) => iL(<>Nt(s(m))\Sf) [iLR]
                                            L(<>EgNt(s(g))\Sf) => <>Nt(s(m))\Sf [\R]
                                                <>Nt(s(m)), L(<>EgNt(s(g))\Sf) => Sf [<>L]
                                                    [Nt(s(m))], L(<>EgNt(s(g))\Sf) => Sf [LL]
                                                        [Nt(s(m))], <>EgNt(s(g))\Sf => Sf [\L]
                                                            [Nt(s(m))] => <>EgNt(s(g)) [<>R]
                                                                Nt(s(m)) => EgNt(s(g)) [ER]
                                                                    Nt(s(m)) => Nt(s(m))
                                                            Sf => Sf
                                        [Nt(s(m))], [[L(<>EgNt(s(g))\Sf), ?iL(<>Nt(s(m))\Sf)\[]-1[]-1(<>Nt(s(m))\Sf)]] => Sf [\L]
                                            L(<>EgNt(s(g))\Sf) => ?iL(<>Nt(s(m))\Sf) [?R]
                                                L(<>EgNt(s(g))\Sf) => iL(<>Nt(s(m))\Sf) [iLR]
                                                    L(<>EgNt(s(g))\Sf) => <>Nt(s(m))\Sf [\R]
                                                        <>Nt(s(m)), L(<>EgNt(s(g))\Sf) => Sf [<>L]
                                                            [Nt(s(m))], L(<>EgNt(s(g))\Sf) => Sf [LL]
                                                                [Nt(s(m))], <>EgNt(s(g))\Sf => Sf [\L]
                                                                    [Nt(s(m))] => <>EgNt(s(g)) [<>R]
                                                                        Nt(s(m)) => EgNt(s(g)) [ER]
                                                                            Nt(s(m)) => Nt(s(m))
                                                                    Sf => Sf
                                                [Nt(s(m))], [[[]-1[]-1(<>Nt(s(m))\Sf)]] => Sf [[]-1L]
                                                    [Nt(s(m))], [[]-1(<>Nt(s(m))\Sf)] => Sf [[]-1L]
                                                        [Nt(s(m))], <>Nt(s(m))\Sf => Sf [\L]
                                                            [Nt(s(m))] => <>Nt(s(m)) [<>R]
                                                                Nt(s(m)) => Nt(s(m))
                                                            Sf => Sf
                    Sf => Sf

AC[(vman C) -> [(Pres (vwalk C)) v (Pres (vtalk C))]]


.
.
.
```

Part III

Appendix B. Corpus of Examples: LaTeX output

Consider consulting f9.1.1.pl and then executing the queries:

```
?- pplexlatex, t(_).
```

Invoking LaTeX on out.tex yields here:

007 : $\blacksquare\forall g Nt(s(g))$: *007*
a : $\blacksquare\forall g(\forall f((Sf\uparrow\blacksquare Nt(s(g)))\downarrow Sf)/CNs(g))$: $\lambda A\lambda B\exists C[(A\ C)\wedge(B\ C)]$
admire : $\square(((\diamond)(\exists a Na\backslash\exists g Nt(s(g)))\backslash Sf)/\exists a Na)$: $\hat{}\lambda A\lambda B(Pres\ (\check{}admire\ A)\ B))$
And : $\blacksquare\forall f(Sf/Sf)$: λAA
and : $\blacksquare\forall f((?\blacksquare Sf\backslash[]^{-1}[]^{-1}Sf)/\blacksquare Sf)$: $(\Phi^{n+}\ 0\ and)$
and : $\blacksquare\forall a\forall f((?\blacksquare(\diamond Na\backslash Sf)\backslash[]^{-1}[]^{-1}(\diamond Na\backslash Sf))/\blacksquare(\diamond Na\backslash Sf))$: $(\Phi^{n+}\ (s\ 0)\ and)$
and : $\blacksquare\forall a\forall f((?\blacksquare(Sf/!Na)\backslash[]^{-1}[]^{-1}(Sf/!Na))/\blacksquare(Sf/!Na))$: $(\Phi^{n+}\ (s\ 0)\ and)$
and : $\blacksquare\forall f((?\blacksquare(Sf/\exists aNa)\backslash[]^{-1}[]^{-1}(Sf/\exists aNa))/\blacksquare(Sf/\exists aNa))$: $(\Phi^{n+}\ (s\ 0)\ and)$
and : $\blacksquare\forall w\forall a\forall b\forall f((\blacksquare((Sf\uparrow((((\diamond)Na\backslash Sf)\circ\!\!-Ww)/Nb))\overset{\circ}{\downarrow}Ww)\backslash[]^{-1}[]^{-1}((Sf\uparrow((((\diamond)Na\backslash Sf)\circ\!\!-Ww)/Nb))\overset{\circ}{\downarrow}Ww))/\hat{}\blacksquare((Sf\uparrow((((\diamond)Na\backslash Sf)\circ\!\!-Ww)/Nb))\overset{\circ}{\downarrow}Ww))$: $\lambda A\lambda B\lambda C[(B\ C)\wedge(A\ C)]$
and : $\blacksquare\forall f\forall a((?\blacksquare(((\diamond)Na\backslash Sf)/\exists bNb)\backslash[]^{-1}[]^{-1}(((\diamond)Na\backslash Sf)/\exists bNb))/\blacksquare(((\diamond)Na\backslash Sf)/\exists bNb))$: $(\Phi^{n+}\ (s\ (s\ 0))\ and)$
and : $\blacksquare\forall f\forall a((?\blacksquare(((\diamond)Na\backslash Sf)/\exists bNb)\backslash(\diamond)Na\backslash Sf))\backslash[]^{-1}[]^{-1}((((\diamond)Na\backslash Sf)/\exists bNb)\backslash(\diamond)Na\backslash Sf)))/\blacksquare((((\diamond)Na\backslash Sf)/\exists bNb)\backslash(\diamond)Na\backslash Sf)))$: $(\Phi^{n+}\ (s\ (s\ 0))\ and)$
and : $\blacksquare\forall f\forall a((\blacksquare(((((\diamond)Na\backslash Sf)/(\exists bNb\oplus\exists g((CNg/CNg)\sqcup(CNg\backslash CNg))))\backslash(\diamond)Na\backslash Sf))\backslash[]^{-1}[]^{-1}((((\diamond)Na\backslash Sf)/(\exists bNb\oplus\exists g((CNg/CNg)\sqcup(CNg\backslash CNg))))\backslash(\diamond)Na\backslash Sf)))/\blacksquare((((\diamond)Na\backslash Sf)/(\exists bNb\oplus\exists g((CNg/CNg)\sqcup(CNg\backslash CNg))))\backslash(\diamond)Na\backslash Sf)))$: $\lambda A\lambda B\lambda C\lambda D[((B\ C)\ D)\wedge((A\ C)\ D)]$
and : $\blacksquare\forall a\forall b\forall f((?\blacksquare((((\diamond)Na\backslash Sf)/(\exists cNc\oplus CPb))\backslash(\diamond)Na\backslash Sf))\backslash[]^{-1}[]^{-1}((((\diamond)Na\backslash Sf)/(\exists cNc\oplus CPb))\backslash(\diamond)Na\backslash Sf)))/\blacksquare((((\diamond)Na\backslash Sf)/(\exists cNc\oplus CPb))\backslash(\diamond)Na\backslash Sf)))$: $(\Phi^{n+}\ (s\ (s\ 0))\ and)$
and : $\blacksquare\forall a\forall b\forall f((?\blacksquare(((\diamond)Na\backslash Sf)/PPb)\backslash[]^{-1}[]^{-1}(((\diamond)Na\backslash Sf)/PPb))/\blacksquare(((\diamond)Na\backslash Sf)/PPb))$: $(\Phi^{n+}\ (s\ (s\ 0))\ and)$
and : $\blacksquare\forall a\forall b\forall f((?\blacksquare((((\diamond)Na\backslash Sf)/(\exists cNc\bullet PPb))\backslash(\diamond)Na\backslash Sf))\backslash[]^{-1}[]^{-1}((((\diamond)Na\backslash Sf)/(\exists cNc\bullet PPb))\backslash(\diamond)Na\backslash Sf)))/\blacksquare((((\diamond)Na\backslash Sf)/(\exists cNc\bullet PPb))\backslash(\diamond)Na\backslash Sf)))$: $(\Phi^{n+}\ (s\ (s\ 0))\ and)$
and : $\blacksquare\forall a\forall b\forall f((?\blacksquare(((\diamond)Na\backslash Sf)/!Nb)\backslash[]^{-1}[]^{-1}(((\diamond)Na\backslash Sf)/!Nb))/\blacksquare(((\diamond)Na\backslash Sf)/!Nb))$: $(\Phi^{n+}\ (s\ (s\ 0))\ and)$
ate : $\square(((\diamond)\exists aNa\backslash Sf)/\exists aNa)$: $\hat{}\lambda A\lambda B(Past\ ((\check{}eat\ A)\ B))$
bagels : $\square(Nt(p(n))\&CNp(n))$: $\hat{}((gen\ \check{}bagels),\check{}bagels)$
barn : $\square CNs(n)$: *barn*
be : $\square(((\diamond)W[there]\multimap Sb)/\exists aNa)$: $\hat{}\lambda A(\check{}be\ A)$
before : $\blacksquare(\forall a\forall f(((\diamond)Na\backslash Sf)\backslash((\diamond)Na\backslash Sf))/Sf)$: $\lambda A\lambda B\lambda C((before\ A)\ (B\ C))$
beginning : $\square CNs(n)$: *beginning*
believes : $\square(((\diamond)\exists g Nt(s(g))\backslash Sf)/(CPthat\sqcup\square Sf))$: $\hat{}\lambda A\lambda B(Pres\ ((\check{}believe\ A)\ B))$
bill : $\blacksquare Nt(s(m))$: *b*
bond : $\blacksquare Nt(s(m))$: *b*
book : $\square CNs(n)$: *book*
bought : $\square(((\diamond)\exists aNa\backslash Sf)/(\exists aNa\bullet\exists aNa))$: $\hat{}\lambda A\lambda B(Past\ (((\check{}buy\ \pi_1 A)\ \pi_2 A)\ B))$
bought : $\square(((\diamond)\exists aNa\backslash Sf)/\exists aNa)$: $\hat{}\lambda A\lambda B(Past\ ((\check{}buy\ A)\ B))$
by : $\blacksquare\forall a(((\diamond)Na\backslash S-)\backslash((\diamond)Na\backslash S-))/Na)$: $\lambda A\lambda B\lambda C[[C=A]\wedge(B\ C)]$
by : $\square(\forall n(CNn\backslash CNn)/\exists aNa)$: $\hat{}\lambda A\lambda B((\check{}by\ A)\ B)$
buys : $\square(((\diamond)\exists g Nt(s(g))\backslash Sf)/(\exists aNa\bullet\exists aNa))$: $\hat{}\lambda A\lambda B(Pres\ (((\check{}buy\ \pi_1 A)\ \pi_2 A)\ B))$
calls : $\square(((\diamond)\exists g Nt(s(g))\backslash Sf)/\exists a((W[up]\blacklozenge Na)\sqcup(Na\blacklozenge W[up])))$: $\hat{}\lambda A\lambda B((\check{}phone\ A)\ B)$
catch : $\square(((\diamond)\exists aNa\backslash Sb)/\exists aNa)$: $\hat{}\lambda A\lambda B((\check{}catch\ A)\ B)$
cezanne : $\blacksquare Nt(s(m))$: *c*
cd : $\square CNs(n)$: *cd*
charles : $\blacksquare Nt(s(m))$: *c*
clark : $\blacksquare\forall g Nt(s(g))$: *c*
coffee : $\square(Nt(s(n))\&CNs(n))$: $\hat{}((gen\ \check{}coffee),\check{}coffee)$
created : $\square(((\diamond)\exists aNa\backslash Sf)/\exists aNa)$: $\hat{}\lambda A\lambda B(Past\ ((\check{}create\ A)\ B))$
darkness : $\square(CNs(n)\&Nt(s(n)))$: $\hat{}(\check{}darkness,(gen\ \check{}darkness))$
deep : $\square CNs(n)$: *deep*
did : $\blacksquare\forall a\forall g\forall b\forall h(((((\diamond)Na\backslash Sg)\uparrow(\diamond)Nb\backslash Sh))/(\exists c(\diamond)Nc\backslash Sf))\backslash(((\diamond)Na\backslash Sg)\uparrow(\diamond)Nb\backslash Sh)))$: $\lambda A\lambda B((A\ B)\ B)$
did+too : $((((\diamond)NA\backslash SB)\uparrow(\diamond)NC\backslash SD))/((\diamond)NE\backslash SF))\backslash(((\diamond)NG\backslash SH)\uparrow(NI\backslash SJ))$: $\lambda K\lambda L((K\ L)\ L)$
doesnt : $\blacksquare\forall g\forall a((Sg\uparrow(((\diamond)Na\backslash Sf)/((\diamond)Na\backslash Sb)))\downarrow Sg)$: $\lambda A\neg(A\ \lambda B\lambda C(B\ C))$
dog : $\square CNs(n)$: *dog*
donuts : $\square(Nt(p(n))\&CNp(n))$: $\hat{}((gen\ \check{}donuts),\check{}donuts)$
earth : $\square CNs(n)$: *earth*
eat : $\square(((\diamond)\exists aNa\backslash Sb)/\exists aNa)$: $\hat{}\lambda A\lambda B((\check{}eat\ A)\ B)$
edinburgh : $\blacksquare Nt(s(n))$: *e*
editor : $\square(\forall g CNs(g)/PPof)$: *editor*
every : $\blacksquare\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g))$: $\lambda A\lambda B\forall C[(A\ C)\rightarrow(B\ C)]$

everyone : $\Box\forall f((Sf\uparrow\forall gNt(g))\downarrow Sf) : \hat{}\lambda A\forall B[(\check{}person\ B) \rightarrow (A\ B)]$
face : $\Box CNs(n) : face$
fell : $\Box(\exists a\langle\rangle Na\backslash Sf) : \hat{}\lambda A(Past\ (\check{}fall\ A))$
filed : $\Box((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Past\ ((\check{}file\ A)\ B))$
finds : $\Box((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}find\ A)\ B))$
fish : $\Box CNs(n) : fish$
for : $\blacksquare(PPfor/\exists aNa) : \lambda AA$
form : $\Box(CNs(n)\&Nt(s(n))) : \hat{}(\check{}form, (gen\ \check{}form))$
fortunately : $\Box\forall f(\check{}Sf\downarrow Sf) : fortunately$
friends : $\Box(CNp/PPof) : friends$
from : $\Box((\forall a\forall f((\langle\rangle Na\backslash Sf)\backslash(\langle\rangle Na\backslash Sf))\&\forall n(CNn\backslash CNn))/\exists bNb) : \hat{}\lambda A((\check{}fromadv\ A), (\check{}fromadn\ A))$
gave : $\Box((\langle\rangle\exists aNa\backslash Sf)/(\exists bNb\bullet PPto)) : \hat{}\lambda A\lambda B(Past\ (((\check{}give\ \pi_2 A)\ \pi_1 A)\ B))$
gave : $\Box((\langle\rangle\exists gNt(s(g))\backslash Sf)/(\exists aNa\blacktriangleright W[the, cold, shoulder])) : \hat{}\lambda A\lambda B(Past\ ((\check{}shun\ A)\ B))$
gave : $\Box(((\langle\rangle\exists aNa\backslash Sf)/\exists aNa)/\exists aNa) : \hat{}\lambda A\lambda B\lambda C(Past\ (((\check{}give\ A)\ B)\ C))$
girl : $\Box CNs(f) : girl$
gives : $\Box((\langle\rangle\exists gNt(s(g))\backslash Sf)/(\exists aNa\blacktriangleright W[the, cold, shoulder])) : \hat{}\lambda A\lambda B(Pres\ ((\check{}shun\ A)\ B))$
God : $\blacksquare Nt(s(m)) : God$
good : $\Box\forall n(CNn/CNn) : good$
has : $\Box(((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}have\ A)\ B))$
he : $\blacksquare[]^{-1}\forall g((\blacksquare Sg|\blacksquare Nt(s(m)))/(\langle\rangle Nt(s(m))\backslash Sg)) : \lambda AA$
heaven : $\Box CNs(n) : heaven$
her : $\blacksquare\forall g\forall a((((\langle\rangle Na\backslash Sg)\uparrow\blacksquare Nt(s(f)))\downarrow(\blacksquare(\langle\rangle Na\backslash Sg)|\blacksquare Nt(s(f)))) : \lambda AA$
himself : $\blacksquare\forall f((((\langle\rangle Nt(s(m))\backslash Sf)\uparrow Nt(s(m)))\downarrow(\langle\rangle Nt(s(m))\backslash Sf)) : \lambda A\lambda B((A\ B)\ B)$
horse : $\Box CNs(n) : horse$
in : $\Box(\forall a\forall f((\langle\rangle Na\backslash Sf)\backslash(\langle\rangle Na\backslash Sf))/\exists aNa) : \hat{}\lambda A\lambda B\lambda C((\check{}in\ A)\ (B\ C))$
in : $\Box(\forall f(Sf\div Sf)/\exists aNa) : in$
is : $\blacksquare((\langle\rangle\exists gNt(s(g))\backslash Sf)/(\exists aNa\oplus(\exists g((CNg/CNg)\sqcup(CNg\backslash CNg))-I))) : \lambda A\lambda B(Pres\ (A \rightarrow C.[B = C]; D.((D\ \lambda E[E = B])\ B)))$
it : $\blacksquare W[it] : 0$
it : $\blacksquare\forall f\forall a((((\langle\rangle Na\backslash Sf)\uparrow\blacksquare Nt(s(n)))\downarrow(\blacksquare(\langle\rangle Na\backslash Sf)|\blacksquare Nt(s(n)))) : \lambda AA$
it : $\blacksquare[]^{-1}\forall f((\blacksquare Sf|\blacksquare Nt(s(n)))/(\langle\rangle Nt(s(n))\backslash Sf)) : \lambda AA$
jogs : $\Box(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}jog\ A))$
john : $\blacksquare Nt(s(m)) : j$
laughs : $\Box(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}laugh\ A))$
left : $\Box(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}leave\ A))$
let : $\Box(Sim/Sb) : let$
light : $\Box(CNs(n)\&Nt(s(n))) : \hat{}(\check{}light, (gen\ \check{}light))$
likes : $\Box((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}like\ A)\ B))$
logic : $\Box(Nt(s(n))\&CNs(n)) : \hat{}((gen\ \check{}logic), \check{}logic)$
london : $\blacksquare Nt(s(n)) : l$
loses : $\Box(((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}lose\ A)\ B))$
love : $\Box((\langle\rangle\exists aNa\backslash Sb)/\exists aNa) : \hat{}\lambda A\lambda B((\check{}love\ A)\ B)$
loved : $\Box\forall a\forall b((((\langle\rangle Na\backslash S-)\uparrow Nb)\circledcirc((((\langle\rangle Na\backslash S-)\uparrow Nb)\downarrow\forall g(CNg\backslash CNg))) : \hat{}(\check{}love, \lambda A\lambda B\lambda C[(B\ C) \wedge \exists D((A\ C)\ D)])$
loves : $\Box(((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}love\ A)\ B))$
man : $\Box CNs(m) : man$
mary : $\blacksquare Nt(s(f)) : m$
met : $\Box(((\langle\rangle\exists aNa\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Past\ ((\check{}meet\ A)\ B))$
more : $\blacksquare\forall h\forall g\forall f((Sf\uparrow(((Sh\uparrow Nt(p(g)))\downarrow Sh)/CNp(g)))\downarrow(Sf/\hat{}(CPthan\uparrow\blacksquare(((Sh\uparrow Nt(p(g)))\downarrow Sh)/CNp(g))))) : \lambda A\lambda B[|\lambda C(A\ \lambda D\lambda E[(D\ C) \wedge (E\ C)])| > |\lambda\check{F}(B\ \lambda G\lambda H[(G\ F) \wedge (H\ F)])|]$
mountain : $\Box CNs(n) : mountain$
moved : $\Box(\langle\rangle\exists aNa\backslash Sf) : \hat{}\lambda A(Past\ (\check{}move\ A))$
necessarily : $\blacksquare(SA/\Box SA) : Nec$
of : $\Box((\forall n(CNn\backslash CNn)/\blacksquare\exists bNb)\&(PPof/\exists aNa)) : \hat{}(\check{}of, \lambda AA)$
or : $\blacksquare\forall f((?\blacksquare Sf\backslash[]^{-1}[]^{-1}Sf)/\blacksquare Sf) : (\Phi^{n+}\ 0\ or)$

or : $\blacksquare\forall a\forall f((?\blacksquare(\langle\rangle Na\backslash Sf)\backslash[]^{-1}[]^{-1}(\langle\rangle Na\backslash Sf))/\blacksquare(\langle\rangle Na\backslash Sf)) : (\Phi^{n+}\ (s\ 0)\ or)$
or : $\blacksquare\forall f((?\blacksquare(Sf/(\langle\rangle\exists gNt(s(g))\backslash Sf))\backslash[]^{-1}[]^{-1}(Sf/(\langle\rangle\exists gNt(s(g))\backslash Sf)))/\blacksquare(Sf/(\langle\rangle\exists gNt(s(g))\backslash Sf))) : (\Phi^{n+}\ (s\ 0)\ or)$
or : $\blacksquare\forall a\forall f((?\blacksquare((((\langle\rangle Na\backslash Sf)/\exists bNb)/\exists bNb)\backslash[]^{-1}[]^{-1}((((\langle\rangle Na\backslash Sf)/\exists bNb)/\exists bNb))/\blacksquare((((\langle\rangle Na\backslash Sf)/\exists bNb)/\exists bNb)) : (\Phi^{n+}\ (s\ (s\ (s\ 0)))\ or)$
painting : $\square(CNs(n)/PPof) : \hat{}\lambda A((\check{}of\ A)\ \check{}painting)$
paper : $\square CNs(n) : paper$
park : $\square CNs(n) : park$
past : $\square\forall a\forall f((((\langle\rangle Na\backslash Sf)\backslash(\langle\rangle Na\backslash Sf))/\exists bNb) : \hat{}\lambda A\lambda B\lambda C((\check{}past\ A)\ (B\ C))$
perseverance : $\square(Nt(s(n))\&CNs(n)) : \hat{}((gen\ \check{}perseverance), \check{}perseverance)$
peter : $\blacksquare Nt(s(m)) : p$
phonetics : $\square(Nt(s(n))\&CNs(n)) : \hat{}((gen\ \check{}phonetics), \check{}phonetics)$
praises : $\square(((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}praise\ A)\ B))$
raced : $\square(\langle\rangle\exists aNa\backslash Sf) : \hat{}\lambda A(Past\ (\check{}race\ A))$
raced : $\square\forall a\forall b((((\langle\rangle Na\backslash S-)\uparrow Nb)\circledcirc((((\langle\rangle Na\backslash S-)\uparrow Nb)\downarrow\forall g(CNg\backslash CNg))) : \hat{}(\check{}race2, \lambda A\lambda B\lambda C[(B\ C)\wedge\exists D((A\ C)\ D)])$
rains : $\square(\langle\rangle W[it]\multimap Sf) : \hat{}(Pres\ \check{}itrains)$
reading : $\square(((\langle\rangle\exists aNa\backslash Spsp)/\exists aNa) : \hat{}\lambda A\lambda B((\check{}read\ A)\ B)$
robin : $\blacksquare\forall gNt(s(g)) : r$
said : $\square(((\langle\rangle\exists aNa\backslash Sf)/Sim) : \hat{}\lambda A\lambda B(Past\ ((\check{}say\ A)\ B))$
saw : $\square(((\langle\rangle\exists aNa\backslash Sf)/(\exists aNa\oplus CPthat)) : \hat{}\lambda A\lambda B(Past\ ((A\rightarrow C.(\check{}seee\ C); D.(\check{}seet\ D))\ B))$
seeks : $\square(((\langle\rangle\exists gNt(s(g))\backslash Sf)/\square\forall a\forall f(((Na\backslash Sf)/\exists bNb)\backslash(Na\backslash Sf))) : \hat{}\lambda A\lambda B((\check{}tries\ \hat{}((\check{}A\ \check{}find)\ B))\ B)$
sees : $\square(((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}see\ A)\ B))$
sent : $\square(((\langle\rangle\exists aNa\backslash Sf)/(\exists bNb\bullet PPto)) : \hat{}\lambda A\lambda B(Past\ (((\check{}sent\ \pi_2 A)\ \pi_1 A)\ B))$
sent : $\square((((\langle\rangle\exists aNa\backslash Sf)/\exists aNa)/\exists aNa) : \hat{}\lambda A\lambda B\lambda C(Past\ (((\check{}send\ A)\ B)\ C))$
she : $\blacksquare[]^{-1}\forall g((\blacksquare Sg|\blacksquare Nt(s(f)))/(\langle\rangle Nt(s(f))\backslash Sg)) : \lambda AA$
sings : $\square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}sing\ A))$
slept : $\square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Past\ (\check{}sleep\ A))$
slowly : $\square\forall a\forall f(\square(\langle\rangle Na\backslash Sf)\backslash(\langle\rangle\square Na\backslash Sf)) : \hat{}\lambda A\lambda B(\check{}slowly\ \hat{}(\check{}A\ \check{}B))$
sneezed : $\square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Past\ (\check{}sneeze\ A))$
sold : $\square(((\langle\rangle\exists aNa\backslash Sf)/(\exists bNb\bullet PPfor)) : \hat{}\lambda A\lambda B(Past\ (((\check{}sell\ \pi_2 A)\ \pi_1 A)\ B))$
someone : $\square\forall f((Sf\uparrow\blacksquare\forall gNt(g))\downarrow Sf) : \hat{}\lambda A\exists B[(\check{}person\ B)\wedge(A\ B)]$
Spirit : $\square CNs(m) : Spirit$
studies : $\square(((\langle\rangle\exists gNt(s(g))\backslash Sf)/\exists aNa) : \hat{}\lambda A\lambda B(Pres\ ((\check{}study\ A)\ B))$
such+that : $\blacksquare\forall n((CNn\backslash CNn)/(Sf|\blacksquare Nt(n))) : \lambda A\lambda B\lambda C[(B\ C)\wedge(A\ C)]$
suzy : $\blacksquare Nt(s(f)) : s$
talks : $\square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}talk\ A))$
tall : $\square\forall g(CNg/CNg) : tall$
teetotal : $\square\forall n(CNn/CNn) : \hat{}\lambda A\lambda B[(A\ B)\wedge(\check{}teetotal\ B)]$
tenmilliondollars : $\square Nt(s(n)) : tenmilliondollars$
than : $\blacksquare(CPthan/\square Sf) : \lambda AA$
that : $\blacksquare(CPthat/\square Sf) : \lambda AA$
that : $\blacksquare\forall n([]^{-1}[]^{-1}(CNn\backslash CNn)/\blacksquare(((\langle\rangle Nt(n)\sqcap!\blacksquare Nt(n))\backslash Sf)) : \lambda A\lambda B\lambda C[(B\ C)\wedge(A\ C)]$
the : $\blacksquare\forall n(Nt(n)/CNn) : \iota$
the+cold+shoulder : $\blacksquare W[the, cold, shoulder] : 0$
there : $\blacksquare W[there] : 0$
thinks : $\square(((\langle\rangle\exists gNt(s(g))\backslash Sf)/(CPthat\sqcup\square Sf)) : \hat{}\lambda A\lambda B(Pres\ ((\check{}think\ A)\ B))$
to : $\blacksquare((PPto/\exists aNa)\sqcap\forall n((\langle\rangle Nn\backslash Si)/(\langle\rangle Nn\backslash Sb))) : \lambda AA$
today : $\square\forall a\forall f(((\langle\rangle Na\backslash Sf)\backslash(\langle\rangle Na\backslash Sf)) : \hat{}\lambda A\lambda B(\check{}today\ (A\ B))$
tries : $\square(((\langle\rangle\exists gNt(s(g))\backslash Sf)/\square(\langle\rangle\exists gNt(s(g))\backslash Si)) : \hat{}\lambda A\lambda B((\check{}tries\ \hat{}(\check{}A\ B))\ B)$
unicorn : $\square CNs(n) : unicorn$
up : $\blacksquare W[up] : 0$
upon : $\square((\forall b\forall f((\langle\rangle Nb\backslash Sf)\backslash(\langle\rangle Nb\backslash Sf))\&\forall g(CNg\backslash CNg))/\exists aNa) : \hat{}\lambda A((\check{}uponadv\ A), (\check{}uponadn\ A))$
void : $\square\forall g(CNg/CNg) : void$
walk : $\square(\langle\rangle(\exists aNa-\exists gNt(s(g)))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}walk\ A))$

walk : $\Box(\langle\rangle\exists aNa\backslash Sb) : \hat{}\lambda A(\check{}walk\ A)$
walks : $\Box(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}walk\ A))$
was : $\blacksquare((\langle\rangle\exists gNt(s(g))\backslash Sf)/(\exists aNa\oplus(\exists g((CNg/CNg)\sqcup(CNg\backslash CNg))-I))) : \lambda A\lambda B(Past\ (A \rightarrow C.[B = C]; D.((D\ \lambda E[E = B])\ B)))$
was : $\Box((\langle\rangle W[there]\multimap Sf)/\exists aNa) : \hat{}\lambda A(Past\ (\check{}be\ A))$
waters : $\Box CNp(n) : waters$
which : $\blacksquare\forall n\forall m((Nt(n)\uparrow Nt(m))\downarrow([]^{-1}[]^{-1}(CNm\backslash CNm)/\blacksquare((\langle\rangle Nt(n)\sqcap!\blacksquare Nt(n))\backslash Sf))) : \lambda A\lambda B\lambda C\lambda D[(C\ D) \wedge (B\ (A\ D))]$
who : $\blacksquare\forall h\forall n([]^{-1}[]^{-1}(Nt(n)\backslash((Sh\uparrow Nt(n))\downarrow Sh))/\blacksquare((\langle\rangle Nt(n)\sqcap!\blacksquare Nt(n))\backslash Sf)) : \lambda A\lambda B\lambda C[(A\ B) \wedge (C\ B)]$
will : $\blacksquare\forall a((\langle\rangle Na\backslash Sf)/(\langle\rangle Na\backslash Sb)) : \lambda A\lambda B(Fut\ (A\ B))$
without : $\Box(\forall g(CNg\backslash CNg)/\exists aNa) : \hat{}\lambda A\lambda B\lambda C[(B\ C) \wedge \neg((\check{}with\ A)\ C)]$
without : $\blacksquare\forall a\forall f([]^{-1}((\langle\rangle Na\backslash Sf)\backslash(\langle\rangle Na\backslash Sf))/(\langle\rangle Na\backslash Spsp)) : \lambda A\lambda B\lambda C[(B\ C) \wedge \neg(A\ C)]$
woman : $\Box CNs(f) : woman$
yesterday : $\Box\forall a\forall f((\langle\rangle Na\backslash Sf)\backslash(\langle\rangle Na\backslash Sf)) : \hat{}\lambda A\lambda B(\check{}yesterday\ (A\ B))$

(dwp(7-7))) **[john]**+**walks** : Sf

$[\blacksquare Nt(s(m)) : j], \Box(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda A(Pres\ (\check{}walk\ A)) \Rightarrow Sf$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\boxed{Nt(s(m))} \Rightarrow Nt(s(m))}}{\boxed{\blacksquare Nt(s(m))} \Rightarrow Nt(s(m))}\blacksquare L}{\blacksquare Nt(s(m)) \Rightarrow \boxed{\exists gNt(s(g))}}\exists R}{[\blacksquare Nt(s(m))] \Rightarrow \boxed{\langle\rangle\exists gNt(s(g))}}\langle\rangle R \qquad \dfrac{}{\boxed{Sf} \Rightarrow Sf}$$

$$\dfrac{[\blacksquare Nt(s(m))], \boxed{\langle\rangle\exists gNt(s(g))\backslash Sf} \Rightarrow Sf}{[\blacksquare Nt(s(m))], \boxed{\Box(\langle\rangle\exists gNt(s(g))\backslash Sf)} \Rightarrow Sf}\Box L$$

$(Pres\ (\check{}walk\ j))$

(dwp(7-16))) **[every**+**man]**+**talks** : Sf

$[\blacksquare\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g)) : \lambda A\lambda B\forall C[(A\ C) \rightarrow (B\ C)], \Box CNs(m) : man], \Box(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda D(Pres\ (\check{}talk\ D)) \Rightarrow Sf$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{Nt(s(m)) \Rightarrow Nt(s(m))}}{Nt(s(m)) \Rightarrow \exists g Nt(s(g))}\exists R}{[Nt(s(m))] \Rightarrow \langle\rangle \exists g Nt(s(g))}\langle\rangle R \quad \dfrac{}{Sf \Rightarrow Sf}}{[Nt(s(m))], \langle\rangle \exists g Nt(s(g)) \backslash Sf \Rightarrow Sf}\backslash L}{[Nt(s(m))], \Box(\langle\rangle \exists g Nt(s(g)) \backslash Sf) \Rightarrow Sf}\Box L}{[1], \Box(\langle\rangle \exists g Nt(s(g)) \backslash Sf) \Rightarrow Sf \uparrow Nt(s(m))}\uparrow R \quad \dfrac{}{Sf \Rightarrow Sf}}{[(Sf \uparrow Nt(s(m))) \downarrow Sf], \Box(\langle\rangle \exists g Nt(s(g)) \backslash Sf) \Rightarrow Sf}\downarrow L}{[\forall f((Sf \uparrow Nt(s(m))) \downarrow Sf)], \Box(\langle\rangle \exists g Nt(s(g)) \backslash Sf) \Rightarrow Sf}\forall L}{[\forall f((Sf \uparrow Nt(s(m))) \downarrow Sf)/CNs(m)], \Box CNs(m)], \Box(\langle\rangle \exists g Nt(s(g)) \backslash Sf) \Rightarrow Sf}/L}{[\forall g(\forall f((Sf \uparrow Nt(s(g))) \downarrow Sf)/CNs(g))], \Box CNs(m)], \Box(\langle\rangle \exists g Nt(s(g)) \backslash Sf) \Rightarrow Sf}\forall L}{[\blacksquare \forall g(\forall f((Sf \uparrow Nt(s(g))) \downarrow Sf)/CNs(g))], \Box CNs(m)], \Box(\langle\rangle \exists g Nt(s(g)) \backslash Sf) \Rightarrow Sf}\blacksquare L$$

$$\dfrac{\dfrac{}{CNs(m) \Rightarrow CNs(m)}}{\Box CNs(m) \Rightarrow CNs(m)}\Box L$$

$\forall C[(\check{}man\ C) \rightarrow (Pres\ (\check{}talk\ C))]$

(dwp(7-19))) **[the+fish]+walks** : Sf

$[\blacksquare \forall n(Nt(n)/CNn) : \iota, \Box CNs(n) : fish], \Box(\langle\rangle \exists g Nt(s(g)) \backslash Sf) : \hat{}\lambda A(Pres\ (\check{}walk\ A)) \Rightarrow Sf$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{CNs(n) \Rightarrow CNs(n)}}{\Box CNs(n) \Rightarrow CNs(n)}\Box L \quad \dfrac{}{Nt(s(n)) \Rightarrow Nt(s(n))}}{Nt(s(n))/CNs(n)], \Box CNs(n) \Rightarrow Nt(s(n))}/L}{\forall n(Nt(n)/CNn)], \Box CNs(n) \Rightarrow Nt(s(n))}\forall L}{\blacksquare \forall n(Nt(n)/CNn)], \Box CNs(n) \Rightarrow Nt(s(n))}\blacksquare L}{\blacksquare \forall n(Nt(n)/CNn), \Box CNs(n) \Rightarrow \exists g Nt(s(g))}\exists R}{[\blacksquare \forall n(Nt(n)/CNn), \Box CNs(n)] \Rightarrow \langle\rangle \exists g Nt(s(g))}\langle\rangle R \quad \dfrac{}{Sf \Rightarrow Sf}}{[\blacksquare \forall n(Nt(n)/CNn), \Box CNs(n)], \langle\rangle \exists g Nt(s(g)) \backslash Sf \Rightarrow Sf}\backslash L}{[\blacksquare \forall n(Nt(n)/CNn), \Box CNs(n)], \Box(\langle\rangle \exists g Nt(s(g)) \backslash Sf) \Rightarrow Sf}\Box L$$

(*Pres* (ˇ*walk* (ι ˇ*fish*)))

(dwp((7-32))) **[every+man]+[[walks+or+talks]]** : Sf

$[\blacksquare\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g)) : \lambda A\lambda B\forall C[(A\ C) \to (B\ C)], \square CNs(m) : man], [[\square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda D(Pres\ (\check{}walk\ D)), \blacksquare\forall f((?\blacksquare Sf\backslash[]^{-1}[]^{-1}Sf)/\blacksquare Sf) : (\Phi^{n+}\ 0\ or), \square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda E(Pres\ (\check{}talk\ E))]] \Rightarrow Sf$

$[\blacksquare\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g)) : \lambda A\lambda B\forall C[(A\ C) \to (B\ C)], \square CNs(m) : man], [[\square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda D(Pres\ (\check{}walk\ D)), \blacksquare\forall a\forall f((?\blacksquare(\langle\rangle Na\backslash Sf)\backslash[]^{-1}[]^{-1}(\langle\rangle Na\backslash Sf))/\blacksquare(\langle\rangle Na\backslash Sf)) : (\Phi^{n+}\ (s\ 0)\ or), \square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda E(Pres\ (\check{}talk\ E))]] \Rightarrow Sf$

$$Nt(s(m)) \Rightarrow Nt(s(m))$$
$$\exists R:\ Nt(s(m)) \Rightarrow \exists gNt(s(g))$$
$$\langle\rangle R:\ [Nt(s(m))] \Rightarrow \langle\rangle\exists gNt(s(g)) \qquad Sf \Rightarrow Sf$$
$$\backslash L:\ [Nt(s(m))], \langle\rangle\exists gNt(s(g))\backslash Sf \Rightarrow Sf$$
$$\square L:\ [Nt(s(m))], \square(\langle\rangle\exists gNt(s(g))\backslash Sf) \Rightarrow Sf$$
$$\langle\rangle L:\ \langle\rangle Nt(s(m)), \square(\langle\rangle\exists gNt(s(g))\backslash Sf) \Rightarrow Sf$$
$$\backslash R:\ \square(\langle\rangle\exists gNt(s(g))\backslash Sf) \Rightarrow \langle\rangle Nt(s(m))\backslash Sf$$
$$\blacksquare R:\ \square(\langle\rangle\exists gNt(s(g))\backslash Sf) \Rightarrow \blacksquare(\langle\rangle Nt(s(m))\backslash Sf)$$

$$Nt(s(m)) \Rightarrow Nt(s(m))$$
$$\exists R:\ Nt(s(m)) \Rightarrow \exists gNt(s(g))$$
$$\langle\rangle R:\ [Nt(s(m))] \Rightarrow \langle\rangle\exists gNt(s(g)) \qquad Sf \Rightarrow Sf$$
$$\backslash L:\ [Nt(s(m))], \langle\rangle\exists gNt(s(g))\backslash Sf \Rightarrow Sf$$
$$\square L:\ [Nt(s(m))], \square(\langle\rangle\exists gNt(s(g))\backslash Sf) \Rightarrow Sf$$
$$\langle\rangle L:\ \langle\rangle Nt(s(m)), \square(\langle\rangle\exists gNt(s(g))\backslash Sf) \Rightarrow Sf$$
$$\backslash R:\ \square(\langle\rangle\exists gNt(s(g))\backslash Sf) \Rightarrow \langle\rangle Nt(s(m))\backslash Sf$$
$$\blacksquare R:\ \square(\langle\rangle\exists gNt(s(g))\backslash Sf) \Rightarrow \blacksquare(\langle\rangle Nt(s(m))\backslash Sf)$$
$$?R:\ \square(\langle\rangle\exists gNt(s(g))\backslash Sf) \Rightarrow ?\blacksquare(\langle\rangle Nt(s(m))\backslash Sf)$$

$$Nt(s(m)) \Rightarrow Nt(s(m))$$
$$\langle\rangle R:\ [Nt(s(m))] \Rightarrow \langle\rangle Nt(s(m)) \qquad Sf \Rightarrow Sf$$
$$\backslash L:\ [Nt(s(m))], \langle\rangle Nt(s(m))\backslash Sf \Rightarrow Sf$$
$$[]^{-1}L:\ [Nt(s(m))], [[]^{-1}(\langle\rangle Nt(s(m))\backslash Sf)] \Rightarrow Sf$$
$$[]^{-1}L:\ [Nt(s(m))], [[[]^{-1}[]^{-1}(\langle\rangle Nt(s(m))\backslash Sf)]] \Rightarrow Sf$$

$$\backslash L:\ [Nt(s(m))], [[\square(\langle\rangle\exists gNt(s(g))\backslash Sf), ?\blacksquare(\langle\rangle Nt(s(m))\backslash Sf)\backslash[]^{-1}[]^{-1}(\langle\rangle Nt(s(m))\backslash Sf)]] \Rightarrow Sf$$
$$/L:\ [Nt(s(m))], [[\square(\langle\rangle\exists gNt(s(g))\backslash Sf), (?\blacksquare(\langle\rangle Nt(s(m))\backslash Sf)\backslash[]^{-1}[]^{-1}(\langle\rangle Nt(s(m))\backslash Sf))/\blacksquare(\langle\rangle Nt(s(m))\backslash Sf), \square(\langle\rangle\exists gNt(s(g))\backslash Sf)]] \Rightarrow Sf$$
$$\forall L:\ [Nt(s(m))], [[\square(\langle\rangle\exists gNt(s(g))\backslash Sf), \forall f((?\blacksquare(\langle\rangle Nt(s(m))\backslash Sf)\backslash[]^{-1}[]^{-1}(\langle\rangle Nt(s(m))\backslash Sf))/\blacksquare(\langle\rangle Nt(s(m))\backslash Sf)), \square(\langle\rangle\exists gNt(s(g))\backslash Sf)]] \Rightarrow Sf$$
$$\forall L:\ [Nt(s(m))], [[\square(\langle\rangle\exists gNt(s(g))\backslash Sf), \forall a\forall f((?\blacksquare(\langle\rangle Na\backslash Sf)\backslash[]^{-1}[]^{-1}(\langle\rangle Na\backslash Sf))/\blacksquare(\langle\rangle Na\backslash Sf)), \square(\langle\rangle\exists gNt(s(g))\backslash Sf)]] \Rightarrow Sf$$
$$\blacksquare L:\ [Nt(s(m))], [[\square(\langle\rangle\exists gNt(s(g))\backslash Sf), \blacksquare\forall a\forall f((?\blacksquare(\langle\rangle Na\backslash Sf)\backslash[]^{-1}[]^{-1}(\langle\rangle Na\backslash Sf))/\blacksquare(\langle\rangle Na\backslash Sf)), \square(\langle\rangle\exists gNt(s(g))\backslash Sf)]] \Rightarrow Sf$$
$$\uparrow R:\ [1], [[\square(\langle\rangle\exists gNt(s(g))\backslash Sf), \blacksquare\forall a\forall f((?\blacksquare(\langle\rangle Na\backslash Sf)\backslash[]^{-1}[]^{-1}(\langle\rangle Na\backslash Sf))/\blacksquare(\langle\rangle Na\backslash Sf)), \square(\langle\rangle\exists gNt(s(g))\backslash Sf)]] \Rightarrow Sf\uparrow Nt(s(m)) \qquad Sf \Rightarrow Sf$$
$$\downarrow L:\ [(Sf\uparrow Nt(s(m)))\downarrow Sf], [[\square(\langle\rangle\exists gNt(s(g))\backslash Sf), \blacksquare\forall a\forall f((?\blacksquare(\langle\rangle Na\backslash Sf)\backslash[]^{-1}[]^{-1}(\langle\rangle Na\backslash Sf))/\blacksquare(\langle\rangle Na\backslash Sf)), \square(\langle\rangle\exists gNt(s(g))\backslash Sf)]] \Rightarrow Sf$$
$$\forall L:\ [\forall f((Sf\uparrow Nt(s(m)))\downarrow Sf)], [[\square(\langle\rangle\exists gNt(s(g))\backslash Sf), \blacksquare\forall a\forall f((?\blacksquare(\langle\rangle Na\backslash Sf)\backslash[]^{-1}[]^{-1}(\langle\rangle Na\backslash Sf))/\blacksquare(\langle\rangle Na\backslash Sf)), \square(\langle\rangle\exists gNt(s(g))\backslash Sf)]] \Rightarrow Sf$$

$$CNs(m) \Rightarrow CNs(m)$$
$$\square L:\ \square CNs(m) \Rightarrow CNs(m)$$

$$/L:\ [\forall f((Sf\uparrow Nt(s(m)))\downarrow Sf)/CNs(m), \square CNs(m)], [[\square(\langle\rangle\exists gNt(s(g))\backslash Sf), \blacksquare\forall a\forall f((?\blacksquare(\langle\rangle Na\backslash Sf)\backslash[]^{-1}[]^{-1}(\langle\rangle Na\backslash Sf))/\blacksquare(\langle\rangle Na\backslash Sf)), \square(\langle\rangle\exists gNt(s(g))\backslash Sf)]] \Rightarrow Sf$$
$$\forall L:\ [\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g)), \square CNs(m)], [[\square(\langle\rangle\exists gNt(s(g))\backslash Sf), \blacksquare\forall a\forall f((?\blacksquare(\langle\rangle Na\backslash Sf)\backslash[]^{-1}[]^{-1}(\langle\rangle Na\backslash Sf))/\blacksquare(\langle\rangle Na\backslash Sf)), \square(\langle\rangle\exists gNt(s(g))\backslash Sf)]] \Rightarrow Sf$$
$$\blacksquare L:\ [\blacksquare\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g)), \square CNs(m)], [[\square(\langle\rangle\exists gNt(s(g))\backslash Sf), \blacksquare\forall a\forall f((?\blacksquare(\langle\rangle Na\backslash Sf)\backslash[]^{-1}[]^{-1}(\langle\rangle Na\backslash Sf))/\blacksquare(\langle\rangle Na\backslash Sf)), \square(\langle\rangle\exists gNt(s(g))\backslash Sf)]] \Rightarrow Sf$$

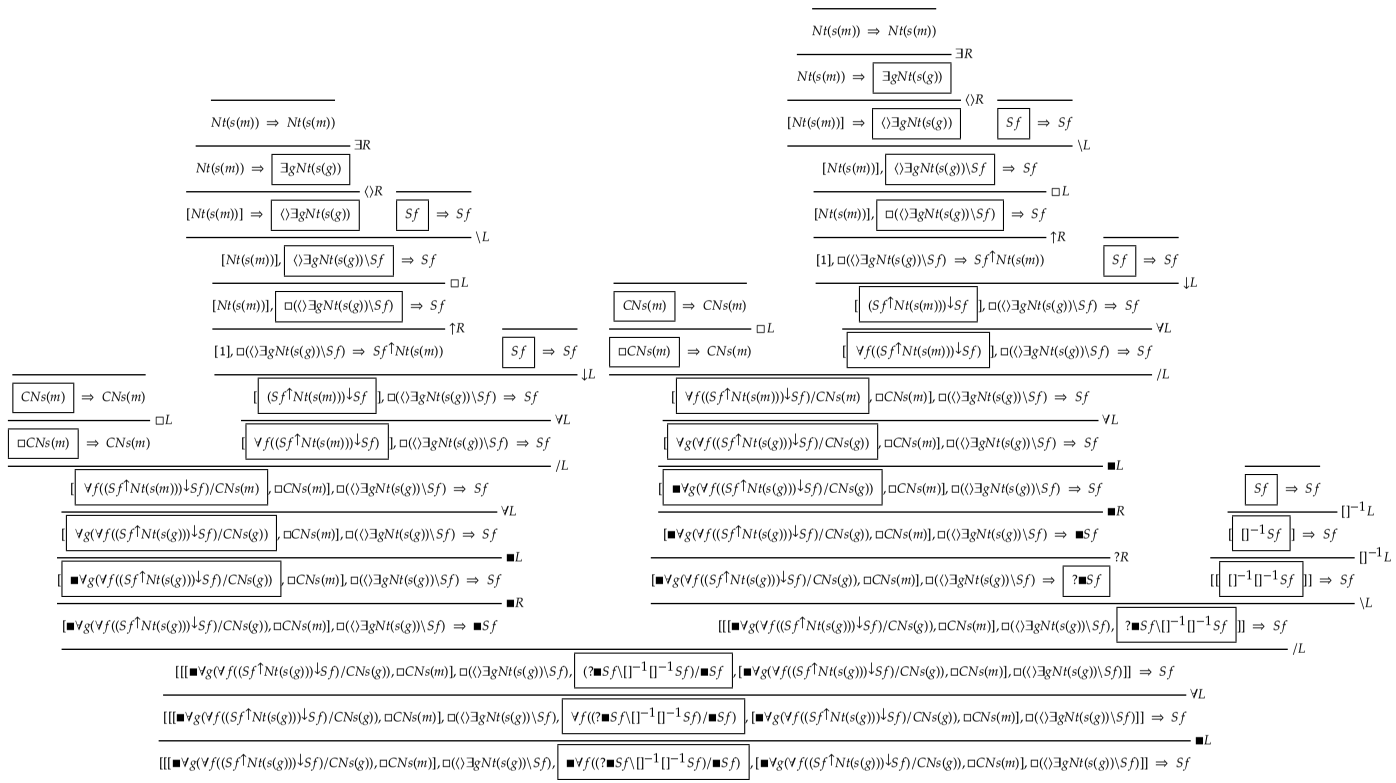
$\forall C[(\check{}man\ C) \to [(Pres\ (\check{}walk\ C)) \vee (Pres\ (\check{}talk\ C))]]$

$[\blacksquare\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g)) : \lambda A\lambda B\forall C[(A\ C) \to (B\ C)], \square CNs(m) : man], [[\square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda D(Pres\ (\check{}walk\ D)), \blacksquare\forall f((?\blacksquare(Sf/(\langle\rangle\exists gNt(s(g))\backslash Sf))\backslash[]^{-1}[]^{-1}(Sf/(\langle\rangle\exists gNt(s(g))\backslash Sf)))/\blacksquare(Sf/(\langle\rangle\exists gNt(s(g))\backslash Sf))) : (\Phi^{n+}\ (s\ 0)\ or), \square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda E(Pres\ (\check{}talk\ E))]] \Rightarrow Sf$

$[\blacksquare\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g)) : \lambda A\lambda B\forall C[(A\ C) \to (B\ C)], \square CNs(m) : man], [[\square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda D(Pres\ (\check{}walk\ D)), \blacksquare\forall a\forall f((?\blacksquare((((\langle\rangle Na\backslash Sf)/\exists bNb)/\exists bNb)\backslash[]^{-1}[]^{-1}((((\langle\rangle Na\backslash Sf)/\exists bNb)/\exists bNb))/\blacksquare((((\langle\rangle Na\backslash Sf)/\exists bNb)/\exists bNb)) : (\Phi^{n+}\ (s\ (s\ (s\ 0)))\ or), \square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda E(Pres\ (\check{}talk\ E))]] \Rightarrow Sf$

(dwp(7-34))) $[[[\textbf{every+man}]+\textbf{walks+or+}[\textbf{every+man}]+\textbf{talks}]] : Sf$

$[[[\blacksquare\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g)) : \lambda A\lambda B\forall C[(A\ C) \to (B\ C)], \square CNs(m) : man], \square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda D(Pres\ (\check{}walk\ D)), \blacksquare\forall f((?\blacksquare Sf\backslash[]^{-1}[]^{-1}Sf)/\blacksquare Sf) : (\Phi^{n+}\ 0\ or), [\blacksquare\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g)) : \lambda E\lambda F\forall G[(E\ G) \to (F\ G)], \square CNs(m) : man], \square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda H(Pres\ (\check{}talk\ H))]] \Rightarrow Sf$



$[\forall H[(\check{}man\ H) \to (Pres\ (\check{}walk\ H))] \vee \forall C[(\check{}man\ C) \to (Pres\ (\check{}talk\ C))]]$

$[[[\blacksquare\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g)) : \lambda A\lambda B\forall C[(A\ C) \rightarrow (B\ C)], \square CNs(m) : man], \square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda D(Pres\ (\check{}walk\ D)), \blacksquare\forall a\forall f((?\blacksquare(\langle\rangle Na\backslash Sf)\backslash[]^{-1}[]^{-1}(\langle\rangle Na\backslash Sf))/\blacksquare(\langle\rangle Na\backslash Sf)) : (\Phi^{n+}\ (s\ 0)\ or), [\blacksquare\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g)) : \lambda E\lambda F\forall G[(E\ G) \rightarrow (F\ G)], \square CNs(m) : man], \square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda H(Pres\ (\check{}talk\ H))]] \Rightarrow Sf$

$[[[\blacksquare\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g)) : \lambda A\lambda B\forall C[(A\ C) \rightarrow (B\ C)], \square CNs(m) : man], \square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda D(Pres\ (\check{}walk\ D)), \blacksquare\forall f((?\blacksquare(Sf/(\langle\rangle\exists gNt(s(g))\backslash Sf))\backslash[]^{-1}[]^{-1}(Sf/(\langle\rangle\exists gNt(s(g))\backslash Sf)))/\blacksquare(Sf/(\langle\rangle\exists gNt(s(g))\backslash Sf))) : (\Phi^{n+}\ (s\ 0)\ or), [\blacksquare\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g)) : \lambda E\lambda F\forall G[(E\ G) \rightarrow (F\ G)], \square CNs(m) : man], \square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda H(Pres\ (\check{}talk\ H))]] \Rightarrow Sf$

$[[[\blacksquare\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g)) : \lambda A\lambda B\forall C[(A\ C) \rightarrow (B\ C)], \square CNs(m) : man], \square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda D(Pres\ (\check{}walk\ D)), \blacksquare\forall a\forall f((?\blacksquare((((\langle\rangle Na\backslash Sf)/\exists bNb)/\exists bNb)\backslash[]^{-1}[]^{-1}((((\langle\rangle Na\backslash Sf)/\exists bNb)/\exists bNb))/\blacksquare((((\langle\rangle Na\backslash Sf)/\exists bNb)/\exists bNb)) : (\Phi^{n+}\ (s\ (s\ (s\ 0)))\ or), [\blacksquare\forall g(\forall f((Sf\uparrow Nt(s(g)))\downarrow Sf)/CNs(g)) : \lambda E\lambda F\forall G[(E\ G) \rightarrow (F\ G)], \square CNs(m) : man], \square(\langle\rangle\exists gNt(s(g))\backslash Sf) : \hat{}\lambda H(Pres\ (\check{}talk\ H))]] \Rightarrow Sf$

(dwp(7-39))) $[[[\mathbf{a}+\mathbf{woman}]+\mathbf{walks}+\mathbf{and}+[\mathbf{she}]+\mathbf{talks}]] : Sf$

.
.
.

Bibliography

[1] J. Lambek. Categorial and Categorical Grammars. In Richard T. Oehrle, Emmon Bach, and Deidre Wheeler, editors, *Categorial Grammars and Natural Language Structures*, volume 32 of *Studies in Linguistics and Philosophy*, pages 297–317. D. Reidel, Dordrecht, 1988.

[2] Michael Moortgat. Multimodal linguistic inference. *Journal of Logic, Language and Information*, 5(3, 4):349–385, 1996. Also in *Bulletin of the IGPL*, 3(2,3):371–401, 1995.

[3] Glyn Morrill. Categorial Formalisation of Relativisation: Pied Piping, Islands, and Extraction Sites. Technical Report LSI-92-23-R, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 1992.

[4] Glyn Morrill and Oriol Valentín. Computational Coverage of TLG: Nonlinearity. In M. Kanazawa, L.S. Moss, and V. de Paiva, editors, *Proceedings of NLCS'15. Third Workshop on Natural Language and Computer Science*, volume 32 of *EPiC*, pages 51–63, Kyoto, 2015. Workshop affiliated with Automata, Languages and Programming (ICALP) and Logic in Computer Science (LICS).

[5] Glyn Morrill, Oriol Valentín, and Mario Fadda. The Displacement Calculus. *Journal of Logic, Language and Information*, 20(1):1–48, 2011.

[6] Glyn V. Morrill. *Categorial Grammar: Logical Syntax, Semantics, and Processing*. Oxford University Press, New York and Oxford, 2011.

[7] Fernando C. N. Pereira and David H. D. Warren. Parsing as deduction. In *Proceedings of 21st Annual Meeting of the Association for Computational Linguistics*, 1983.

[8] J. van Benthem. *Language in Action: Categories, Lambdas, and Dynamic Logic*. Number 130 in Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, 1991. Revised student edition printed in 1995 by the MIT Press.