```
% HPSG GRAMMAR (preliminary version)
%
==========================================================================
% 5 April 1993
% Revised: 11 December 2001

% Gerald Penn, Carnegie Mellon University, gpenn@cmu.edu

% Renamed feature gen to gend for SWI parser
% Added discontiguity declarations for SICStus

:- discontiguous (lex_rule)/2.
:- discontiguous (if)/2.
:- discontiguous ('--->')/2.

% Signature
%
--------------------------------------------------------------------------

bot sub [bool, case, cat, c_inds, conx,
          gend, head, ind, list, loc, marking,
          name, non_loc, non_loc_1, num, mod_synsem,
          per, pform, qfpsoa, sem_det, sem_obj, sign, set, vform].
  bool sub [minus, plus].
    minus sub [].
    plus sub [].
  case sub [nom,acc].
    nom sub [].
    acc sub [].
  cat sub []
      intro [subcat:list_synsem,
             head:head,
             marking:marking].
  c_inds sub []
         intro [addressee:ref,
                speaker:ref,
                utt_loc:ref].
  conx sub []
       intro [backgr:set_psoa,
              c_inds:c_inds].
  gend sub [fem, masc, neut].
    fem sub [].
    masc sub [].
    neut sub [].
  head sub [func, subst].
    func sub [det, mark]
         intro [spec:synsem].
```

```
    det sub [].
    mark sub [].
  subst sub [adj, noun, prep, reltvzr, verb]
        intro [prd:bool,
               mod:mod_synsem].
    adj sub [].
    noun sub []
         intro [case:case].
    prep sub []
         intro [pform:pform].
    reltvzr sub [].
    verb sub []
         intro [aux:bool,
                inv:bool,
                vform:vform].
mod_synsem sub [synsem,none].
  synsem sub []
         intro [loc:loc,
                non_loc:non_loc].
  none sub [].
ind sub [it, there, ref]
    intro [gend:gend,
           num:num,
           per:per].
  it sub [].
  there sub [].
  ref sub [].
list sub [e_list, ne_list, list_quant, list_synsem, list_sign].
  e_list sub [].
  ne_list sub [ne_list_quant,
               ne_list_synsem,
               ne_list_sign]
          intro [hd:bot,
                 tl:list].
  list_quant sub [e_list, ne_list_quant].
    ne_list_quant sub []
                  intro [hd:quant,
                         tl:list_quant].
  list_synsem sub [e_list, ne_list_synsem].
    ne_list_synsem sub []
                  intro [hd:synsem,
                         tl:list_synsem].
  list_sign sub [e_list, ne_list_sign].
    ne_list_sign sub []
                  intro [hd:sign,
                         tl:list_sign].
loc sub []
```

```
    intro [cat:cat,
           cont:sem_obj,
           conx:conx].
marking sub [marked, unmarked].
  marked sub [comp, conj].
    comp sub [for, that].
      for sub [].
      that sub [].
    conj sub [].
  unmarked sub [].
name sub [kim,sandy].
  kim sub [].
  sandy sub [].
non_loc sub []
        intro [inherited:non_loc_1,
               to_bind:non_loc_1].
non_loc_1 sub []
          intro [que:set_npro,
                 rel:set_ref,
                 slash:set_loc].
num sub [plur, sing].
  plur sub [].
  sing sub [].
per sub [first, second, third].
  first sub [].
  second sub [].
  third sub [].
pform sub [to, of].
  to sub [].
  of sub [].
qfpsoa sub [property, un_relation, bin_relation, tri_relation,
            control_qfpsoa].
  property sub [gender,nom_prop]
           intro [inst:ref].
    gender sub [human,neuter].
      human sub [female,male].
        female sub [].
        male sub [].
      neuter sub [].
    nom_prop sub [book,red,difficult].
      book sub [].
      red sub [].
      difficult sub [].
  un_relation sub [walk,run].
    walk sub [] intro [walker:ref].
    run sub [] intro [runner:ref].
  bin_relation sub [see,hit,naming,composed_of,possess].
```

```
      see sub [] intro [seer:ref, seen:ref].
      hit sub [] intro [hitter:ref, hittee:ref].
      naming sub [] intro [bearer:ref, name:name].
      composed_of sub [] intro [composite:ref, composition:set_ref].
      possess sub [] intro [possessor:ref, possessed:ref].
    tri_relation sub [sell,give].
      sell sub [] intro [seller:ref, buyer:ref, sold:ref].
      give sub [] intro [giver:ref, given:ref, gift:ref].
    control_qfpsoa sub [trying, tending, believing, persuading, bothering]
                   intro [soa_arg:psoa].
      trying sub [] intro [tryer:ref].
      persuading sub [] intro [persuader:ref, persuaded:ref].
      tending sub [].
      believing sub [] intro [believer:ref].
      bothering sub [] intro [bothered:ref].
  sem_det sub [forall,exists,the].
    forall sub [].
    exists sub [].
    the sub [].
  sem_obj sub [nom_obj, psoa, quant].
    nom_obj sub [npro, pron]
            intro [index:ind,
                   restr:set_psoa].
      npro sub [].
      pron sub [ana, ppro].
        ana sub [recp, refl].
          recp sub [].
          refl sub [].
        ppro sub [].
    quant sub []
          intro [det:sem_det,
                 restind:npro].
    psoa sub []
         intro [quants:list_quant,nucleus:qfpsoa].
  sign sub [word,non_word]
       intro [synsem:synsem,
              qstore:set_quant,
              qretr:list_quant].
    word sub [].
    non_word sub [trace,phrase].
      trace sub [].
      phrase sub [].
  set sub [e_set, ne_set, set_loc, set_npro, set_psoa, set_quant, set_ref].
    e_set sub [].
    ne_set sub [ne_set_loc, ne_set_npro, ne_set_psoa, ne_set_quant,
ne_set_ref]
           intro [elt:bot, elts:set].
```

```
    set_loc sub [e_set, ne_set_loc].
      ne_set_loc sub []
                 intro [elt:loc, elts:set_loc].
    set_npro sub [e_set, ne_set_npro].
      ne_set_npro sub []
                 intro [elt:npro, elts:set_npro].
    set_psoa sub [e_set, ne_set_psoa].
      ne_set_psoa sub []
                 intro [elt:psoa, elts:set_psoa].
    set_quant sub [e_set, ne_set_quant].
      ne_set_quant sub []
                 intro [elt:quant, elts:set_quant].
    set_ref sub [e_set, ne_set_ref].
      ne_set_ref sub []
                 intro [elt:ref, elts:set_ref].
  vform sub [bse, fin, ger, inf, pas, prp, psp].
    bse sub [].
    fin sub [].
    ger sub [].
    inf sub [].
    pas sub [].
    prp sub [].
    psp sub [].


% Constraints
%
================================================================================

word cons Word
     goal (single_rel_constraint(Word),
           clausal_rel_prohibition(Word)).

trace cons
  synsem:(loc:(cat:(Cat,head:mod:none,
                          subcat:[]),
               cont:Cont),
          non_loc:(inherited:(que:e_set,
                              rel:e_set,
                              slash:(elt:(cat:Cat,
                                          cont:Cont),
                                     elts:e_set)),
                   to_bind:(que:e_set,
                            rel:e_set,
                            slash:e_set))).

% Lexicon
```

```
%
================================================================================

% Personal Pronouns
%
--------------------------------------------------------------------------------

she --->
  word,
  synsem:loc:(cat:(head:(noun,
                         case:nom,
                         mod:none),
                   subcat: [],
                   marking:unmarked),
              cont:(ppro,
                    index:(ref,Ind,
                           per:third,
                           num:sing,
                           gend:fem),
                    restr:e_set),
              conx:backgr:(elt:(nucleus:(female,
                                         inst:Ind),
                                quants:[]),
                           elts:e_set)),
  (@ empty_non_loc),
  qstore:e_set.

he --->
  word,
  synsem:loc:(cat:(head:(noun,
                         case:nom,
                         mod:none),
                   subcat: [],
                   marking:unmarked),
              cont:(ppro,
                    index:(ref,Ind,
                           per:third,
                           num:sing,
                           gend:masc),
                    restr:e_set),
              conx:backgr:(elt:(nucleus:(male,
                                         inst:Ind),
                                quants:[]),
                           elts:e_set)),
  (@ empty_non_loc),
  qstore:e_set.
```

```
her --->
  word,
  synsem:loc:(cat:(head:(noun,
                         case:acc,
                         mod:none),
                   subcat: [],
                   marking:unmarked),
              cont:(ppro,
                    index:(ref,Ind,
                           per:third,
                           num:sing,
                           gend:fem),
                    restr:e_set),
              conx:backgr:(elt:(nucleus:(female,
                                         inst:Ind),
                                quants:[]),
                           elts:e_set)),
  (@ empty_non_loc),
  qstore:e_set.

him --->
  word,
  synsem:loc:(cat:(head:(noun,
                         case:acc,
                         mod:none),
                   subcat: [],
                   marking:unmarked),
              cont:(ppro,
                    index:(ref,Ind,
                           per:third,
                           num:sing,
                           gend:masc),
                    restr:e_set),
              conx:backgr:(elt:(nucleus:(male,
                                         inst:Ind),
                                quants:[]),
                           elts:e_set)),
  (@ empty_non_loc),
  qstore:e_set.

it --->
  word,
  synsem:loc:(cat:(head:(noun,
                         mod:none),
                   subcat: [],
                   marking:unmarked),
              cont:(ppro,
```

```
                    index:(ref,Ind,
                           per:third,
                           num:sing,
                           gend:neut),
                    restr:e_set),
              conx:backgr:(elt:(nucleus:(neuter,
                                         inst:Ind),
                                quants:[]),
                           elts:e_set)),
  (@ empty_non_loc),
  qstore:e_set.

i --->
  word,
  synsem:loc:(cat:(head:(noun,
                           case:nom,
                           mod:none),
                    subcat: [],
                    marking:unmarked),
              cont:(index:(ref,Ind,
                           per:first,
                           num:sing),
                    restr:e_set),
              conx:(backgr:e_set,
                    c_inds:speaker:Ind)),
  (@ empty_non_loc),
  qstore:e_set.

me --->
  word,
  synsem:loc:(cat:(head:(noun,
                           case:acc,
                           mod:none),
                    subcat: [],
                    marking:unmarked),
              cont:(index:(ref,Ind,
                           per:first,
                           num:sing),
                    restr:e_set),
              conx:(backgr:e_set,
                    c_inds:speaker:Ind)),
  (@ empty_non_loc),
  qstore:e_set.

we --->
  word,
  synsem:loc:(cat:(head:(noun,
```

```
                        case:nom,
                        mod:none),
                 subcat: [],
                 marking:unmarked),
          cont:(index:(ref,Ind,
                        gend:neut,
                        per:first,
                        num:plur),
                restr:(elt:(nucleus:(composed_of,
                                     composite:Ind,
                                     composition:(elt:Ind2,
                                                  elts:(elt:Ind3,
                                                        elts:e_set))),
                                quants:[]),
                        elts:e_set)),
          conx:(backgr:e_set,
                c_inds:(speaker:Ind2,
                        addressee:Ind3))),
  (@ empty_non_loc),
  qstore:e_set.

us --->
  word,
  synsem:loc:(cat:(head:(noun,
                        case:acc,
                        mod:none),
                 subcat: [],
                 marking:unmarked),
          cont:(index:(ref,Ind,
                        gend:neut,
                        per:first,
                        num:plur),
                restr:(elt:(nucleus:(composed_of,
                                     composite:Ind,
                                     composition:(elt:Ind2,
                                                  elts:(elt:Ind3,
                                                        elts:e_set))),
                                quants:[]),
                        elts:e_set)),
          conx:(backgr:e_set,
                c_inds:(speaker:Ind2,
                        addressee:Ind3))),
  (@ empty_non_loc),
  qstore:e_set.

you --->
  word,
```

```
  synsem:loc:(cat:(head:(noun,
                         mod:none),
                   subcat: [],
                   marking:unmarked),
              cont:(index:(ref,Ind,
                         per:second),
                    restr:e_set),
              conx:(backgr:e_set,
                    c_inds:addressee:Ind)),
  (@ empty_non_loc),
  qstore:e_set.

they --->
  word,
  synsem:loc:(cat:(head:(noun,
                         case:nom,
                         mod:none),
                   subcat: [],
                   marking:unmarked),
              cont:(ppro,
                    index:(ref,Ind,
                         per:third,
                         num:plur,
                         gend:neut),
                    restr:e_set),
              conx:backgr:(elt:(nucleus:(neuter,
                                         inst:Ind),
                                quants:[]),
                         elts:e_set)),
  (@ empty_non_loc),
  qstore:e_set.

them --->
  word,
  synsem:loc:(cat:(head:(noun,
                         case:acc,
                         mod:none),
                   subcat: [],
                   marking:unmarked),
              cont:(ppro,
                    index:(ref,Ind,
                         per:third,
                         num:plur,
                         gend:neut),
                    restr:e_set),
              conx:backgr:(elt:(nucleus:(neuter,
                                         inst:Ind),
```

```
                                quants:[]),
                           elts:e_set)),
  (@ empty_non_loc),
  qstore:e_set.


% Expletive Pronouns
%
-------------------------------------------------------------------------------

there --->
  word,
  synsem:loc:(cat:(head:(noun,
                         mod:none),
                   subcat:[]),
              cont:(ppro,
                    index:(there,
                           per:third),
                    restr:e_set),
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

it --->
  word,
  synsem:loc:(cat:(head:(noun,
                         mod:none),
                   subcat:[]),
              cont:(ppro,
                    index:(it,
                           per:third,
                           num:sing),
                    restr:e_set),
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

% Relative Pronouns
%
-------------------------------------------------------------------------------

who --->
  word,
  synsem:(loc:(cat:(head:(noun,
                          case:nom,
                          mod:none),
                    subcat:[]),
```

```
               cont:(npro,
                     index:Ind,
                     restr:e_set),
               conx:backgr:(elt:(nucleus:(human,
                                          inst:Ind),
                                quants:[]),
                           elts:e_set)),
         non_loc:(inherited:(que:e_set,
                             rel:(elt:Ind,elts:e_set),
                             slash:e_set),
                  to_bind:(que:e_set,
                           rel:e_set,
                           slash:e_set))),
  qstore:e_set.

whom --->
  word,
  synsem:(loc:(cat:(head:(noun,
                          case:acc,
                          mod:none),
                    subcat:[]),
               cont:(npro,
                     index:Ind,
                     restr:e_set),
               conx:backgr:(elt:(nucleus:(human,
                                          inst:Ind),
                                quants:[]),
                           elts:e_set)),
         non_loc:(inherited:(que:e_set,
                             rel:(elt:Ind,elts:e_set),
                             slash:e_set),
                  to_bind:(que:e_set,
                           rel:e_set,
                           slash:e_set))),
  qstore:e_set.

that --->
  word,
  synsem:(loc:(cat:(head:(noun,
                          case:nom,
                          mod:none),
                    subcat:[]),
               cont:(npro,
                     index:Ind,
                     restr:e_set),
               conx:backgr:e_set),
         non_loc:(inherited:(que:e_set,
```

```
                              rel:(elt:Ind,elts:e_set),
                              slash:e_set),
                    to_bind:(que:e_set,
                             rel:e_set,
                             slash:e_set))),
  qstore:e_set.

% Proper Names
%
------------------------------------------------------------------------------

kim --->
  word,
  synsem:loc:(cat:(head:(noun,
                         mod:none),
                   subcat: [],
                   marking:unmarked),
              cont:(index:(ref,Ind,
                           per:third,
                           num:sing),
                    restr:e_set),
              conx:backgr:(elt:(nucleus:(naming,
                                         bearer:Ind,
                                         name:kim),
                                quants:[]),
                           elts:e_set)),
  (@ empty_non_loc),
  qstore:e_set.

sandy --->
  word,
  synsem:loc:(cat:(head:(noun,
                         mod:none),
                   subcat: [],
                   marking:unmarked),
              cont:(index:(ref,Ind,
                           per:third,
                           num:sing),
                    restr:e_set),
              conx:backgr:(elt:(nucleus:(naming,
                                         bearer:Ind,
                                         name:sandy),
                                quants:[]),
                           elts:e_set)),
  (@ empty_non_loc),
  qstore:e_set.
```

```
% Common Nouns
%
------------------------------------------------------------------------------

book --->
  word,
  synsem:loc:(cat:(head:(noun,
                         mod:none),
                   subcat:[(@ detp)],
                   marking:unmarked),
              cont:(npro,
                    index:(ref,Ind,
                           per:third,
                           num:sing,
                           gend:neut),
                    restr:(elt:(nucleus:(book,
                                         inst:Ind),
                                quants:[]),
                           elts:e_set)),
              conx:backgr:e_set),
  (@ empty_inher),
  qstore:e_set.

person --->
  word,
  synsem:loc:(cat:(head:(noun,
                         mod:none),
                   subcat:[(@ detp)],
                   marking:unmarked),
              cont:(npro,
                    index:(ref,Ind,
                           per:third,
                           num:sing,
                           gend:(masc;fem)),
                    restr:e_set),
              conx:backgr:(elt:(nucleus:(human,
                                         inst:Ind),
                                quants:[]),
                           elts:e_set)),
  (@ empty_inher),
  qstore:e_set.

% Quantificational Determiners
%
------------------------------------------------------------------------------
```

```
every --->
  word,
  synsem:loc:(cat:(head:(det,
                         spec:(@ nbar(NPCont))),
                   subcat:[],
                   marking:unmarked),
              cont:(GQ,
                    det:forall,
                    restind:NPCont),
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:(elt:GQ,
          elts:e_set).

a --->
  word,
  synsem:loc:(cat:(head:(det,
                         spec:(@ nbar(NPCont))),
                   subcat:[],
                   marking:unmarked),
              cont:(GQ,
                    det:exists,
                    restind:NPCont),
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:(elt:GQ,
          elts:e_set).

the --->
  word,
  synsem:loc:(cat:(head:(det,
                         spec:(@ nbar(NPCont))),
                   subcat:[],
                   marking:unmarked),
              cont:(GQ,
                    det:the,
                    restind:NPCont),
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:(elt:GQ,
          elts:e_set).

% Possessive Pronouns
%
-------------------------------------------------------------------------

my --->
```

```
  word,
  synsem:loc:(cat:(head:(det,
                         spec:(@ nbar((index:Ind,
                                       restr:Restr)))),
                  subcat:[],
                  marking:unmarked),
              cont:(index:(Ind2,
                           per:first,
                           num:sing),
                    restr:e_set),
              conx:(c_inds:speaker:Ind2,
                    backgr:e_set)),
  (@ empty_non_loc),
  qstore:(elt:(det:the,
               restind:(index:Ind,
                        restr:(elt:(nucleus:(possess,
                                             possessor:Ind2,
                                             possessed:Ind),
                                    quants:[]),
                                elts:Restr))),
          elts:e_set).


% Possessive Clitic
%
--------------------------------------------------------------------------------

s --->
  word,
  synsem:loc:(cat:(head:(det,
                         spec:(@ nbar((index:Ind,
                                       restr:Restr)))),
                   subcat:[(@ np(Ind2),
                            loc:cont:(npro,NPCont))],
                   marking:unmarked),
              cont:NPCont,
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:(elt:(det:the,
               restind:(index:Ind,
                        restr:(elt:(nucleus:(possess,
                                             possessor:Ind2,
                                             possessed:Ind),
                                    quants:[]),
                                elts:Restr))),
          elts:e_set).
```

```
% Adjectives
%
-------------------------------------------------------------------------

% attributive
%----------------
red --->
  word,
  synsem:loc:(cat:(head:(adj,
                         prd:minus,
                         mod:(@ nbar((index:Ind,
                                      restr:Restrs)))),
                   subcat:[],
                   marking:unmarked),
              cont:(index:Ind,
                    restr:(elt:(nucleus:(red,
                                         inst:Ind),
                                quants:[]),
                           elts:Restrs)),
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

% predicative
%-----------------


% tough
%-----------------


% Verbs
%
-------------------------------------------------------------------------

walk --->
  word,
  synsem:loc:(cat:(head:(verb,            %n.b. prd not specified - it would
                         mod:none,          % pass to sentential complements
                         vform:bse,         % which could then not be used
                         aux:minus,         % predicatively, e.g. w/ there-
                         inv:minus),        % extraposition, pp. 163-173, or
                                            % passive lexical rule (at least
                                            % as it stands now)
                   subcat:[(@ np(Ind))],   %n.b. case not specified for
```

```
                    marking:unmarked),      % nonfinite forms
              cont:(nucleus:(walk,
                             walker:Ind),
                    quants:[]),
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

see --->
  word,
  synsem:loc:(cat:(head:(verb,
                         mod:none,
                         vform:bse,
                         aux:minus,
                         inv:minus),
                   subcat:[ (@ np(Ind1)),
                            (@ np(Ind2), @ case(acc))],
                   marking:unmarked),
              cont:(nucleus:(see,
                             seer:Ind1,
                             seen:Ind2),
                    quants:[]),
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

give --->
  word,
  synsem:loc:(cat:(head:(verb,
                         mod:none,
                         vform:bse,
                         aux:minus,
                         inv:minus),
                   subcat:[ (@ np(Ind1)),
                            (@ np(Ind2), @ case(acc)),
                            (@ np(Ind3), @ case(acc))],
                   marking:unmarked),
              cont:(nucleus:(give,
                             giver:Ind1,
                             given:Ind2,
                             gift:Ind3),
                    quants:[]),
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

bother --->
```

```
  synsem:loc:(cat:(head:(verb,
                         mod:none,
                         vform:bse,
                         aux:minus,
                         inv:minus),
                   subcat:[(@ s(SCont),loc:cat:marking:comp),
                           (@ np(Ind2))],
                   marking:unmarked),
              cont:(nucleus:(bothering,
                             bothered:Ind2,
                             soa_arg:SCont),
                    quants:[]),
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

try --->                              % subject equi
  word,
  synsem:loc:(cat:(head:(verb,
                         mod:none,
                         vform:bse,
                         aux:minus,
                         inv:minus),
                   subcat:[(@ np(Ind1)),
                           (@ vp(VCont),
                            loc:cat:(head:vform:inf,
                                     subcat:[(@ np(Ind1))]))],
                   marking:unmarked),
              cont:(nucleus:(trying,
                             tryer:Ind1,
                             soa_arg:VCont),
                    quants:[]),
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

tend --->                               % subject raising
  word,
  synsem:loc:(cat:(head:(verb,
                         mod:none,
                         vform:bse,
                         aux:minus,
                         inv:minus),
                   subcat:[(loc:Loc,@ np(_)),
                           (@ vp(VCont),
                            loc:cat:(head:vform:inf,
                                     subcat:[(loc:Loc)]))],
```

```
                    marking:unmarked),
              cont:(nucleus:(tending,
                             soa_arg:VCont),
                    quants:[]),
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

persuade --->                              % object equi
  word,
  synsem:loc:(cat:(head:(verb,
                         mod:none,
                         vform:bse,
                         aux:minus,
                         inv:minus),
                   subcat:[(@ np(Ind1)),
                           (@ np(Ind2), @ case(acc)),
                           (@ vp(VCont),
                            loc:cat:(head:vform:inf,
                                     subcat:[(@ np(Ind2))]))],
                   marking:unmarked),
              cont:(nucleus:(persuading,
                             persuader:Ind1,
                             persuaded:Ind2,
                             soa_arg:VCont),
                    quants:[]),
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

believe --->                               % object raising
  word,
  synsem:loc:(cat:(head:(verb,
                         mod:none,
                         vform:bse,
                         aux:minus,
                         inv:minus),
                   subcat:[(@ np(Ind1)),
                           (loc:Loc2,@ np(_), @ case(acc)),
                           (@ vp(VCont),
                            loc:cat:(head:vform:inf,
                                     subcat:[(loc:Loc2)]))],
                   marking:unmarked),
              cont:(nucleus:(believing,
                             believer:Ind1,
                             soa_arg:VCont),
                    quants:[]),
```

```
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

% Auxiliaries
%
-------------------------------------------------------------------------------

can --->
  word,
  synsem:loc:(cat:(head:(verb,
                         mod:none,
                         vform:fin,
                         aux:plus),
                   subcat:[ (NP, @ np(_), @ case(nom)),
                            (@ vp(Prop),
                             loc:cat:(head:vform:bse,
                                      subcat:[NP]))],
                   marking:unmarked),
              cont:Prop,
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

to --->
  word,
  synsem:loc:(cat:(head:(verb,
                         mod:none,
                         vform:inf,
                         aux:plus,
                         inv:minus),
                   subcat:[(NP, @ np(_)),         % n.b. case not specified
                           (@ vp(Prop),           %  for nonfinite forms
                            loc:cat:(head:vform:bse,
                                     subcat:[NP]))],
                   marking:unmarked),
              cont:Prop,
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

be --->                          % ideally, this would be able to handle
  word,                          %  more than passive verb forms as
predicates
  synsem:loc:(cat:(head:(verb,
                         mod:none,
                         vform:bse,
```

```
                          aux:plus),
                    subcat:[(NP,@ np(_)),
                            (loc:(cat:(head:(vform:pas,
                                             aux:minus, % no "be been"
                                             prd:plus),
                                       subcat:[NP]),
                                  cont:Cont))],
                    marking:unmarked),
              cont:Cont,
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

% Complementizers
%
------------------------------------------------------------------------------

that --->
  word,
  synsem:loc:(cat:(head:(mark,
                         spec:(@ s(_),
                               loc:cat:(head:vform:( fin
                                                   ; bse),
                                        marking:unmarked))),
                    subcat:[],
                    marking:that),
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

% Empty Categories
%
==============================================================================

% trace introduction
%--------------------
empty
  trace.

% wh-null-relativzer
%--------------------
empty
  synsem:(loc:(cat:(head:(reltvzr,
                          mod:(@ nbar((index:ModInd,
                                       restr:ModRestr)),
                               non_loc:to_bind:rel:(elt:ModInd,
                                                    elts:e_set))),
```

```
                    subcat:[(loc:SubLoc,
                             non_loc:inherited:rel:(elt:ModInd,
                                                    elts:e_set)),
                            (@ s(SCont),
                             loc:cat:head:vform:fin,
                             non_loc:inherited:slash:(elt:SubLoc,
                                                      elts:e_set))]),
              cont:(npro,
                    index:ModInd,
                    restr:(elt:SCont,
                           elts:ModRestr))),
          non_loc:(inherited:(slash:e_set,
                              que:e_set,
                              rel:e_set),
                   to_bind:(slash:(elt:SubLoc,
                                   elts:e_set),
                            que:e_set,
                            rel:e_set))).

% SELR form of wh-null-relativzer
%--------------------------------
% should be derived by SELR, but cannot until Raising Principle can be
% implemented
empty
  synsem:(loc:(cat:(head:(reltvzr,
                          mod:(@ nbar((index:ModInd,
                                       restr:ModRestr)),
                                 non_loc:to_bind:rel:(elt:ModInd,
                                                      elts:e_set))),
                    subcat:[(@ np(_),
                             loc:SubLoc,
                             non_loc:inherited:rel:(elt:ModInd,
                                                    elts:e_set)),
                            (@ vp(SCont),
                             loc:cat:(head:vform:fin,
                                      subcat:[(loc:SubLoc)]))]),
              cont:(npro,
                    index:ModInd,
                    restr:(elt:SCont,
                           elts:ModRestr))),
          non_loc:(inherited:(slash:(elt:SubLoc,
                                     elts:e_set),
                              que:e_set,
                              rel:e_set),
                   to_bind:(slash:(elt:SubLoc,
                                   elts:e_set),
                            que:e_set,
```

```
                            rel:e_set))).

% that-null-relativizer
%-----------------------
% (note: not subject to SELR)
empty
  synsem:(loc:(cat:(head:(reltvzr,
                            mod:(@ nbar((index:ModInd,
                                         restr:ModRestr)),
                                   non_loc:to_bind:rel:(elt:ModInd,
                                                        elts:e_set))),
                     subcat:[(@ s(SCont),
                              loc:cat:head:vform:fin,
                              non_loc:inherited:slash:(elt:SubSlash,
                                                       elts:e_set))]),
               cont:(npro,
                     index:ModInd,
                     restr:(elt:SCont,
                            elts:ModRestr))),
          non_loc:(inherited:(slash:e_set,
                              que:e_set,
                              rel:(elt:ModInd,
                                   elts:e_set)),
                   to_bind:(slash:(elt:(SubSlash,cat:(head:noun,
                                                      subcat:[]),
                                                cont:index:ModInd),
                                   elts:e_set),
                            que:e_set,
                            rel:e_set))).


% Lexical Rules
%
==========================================================================

% Finite Verb Formation
% ----------------------------------------------------------------------

% regulars: lexical rule
pres_3s lex_rule
  (word,
   synsem:(loc:(cat:(head:(verb,
                           vform:bse,
                           aux:minus,
                           inv:Inv,
                           prd:Prd,
                           mod:Mod),
```

```
                        subcat:[Sub|SubRest],
                        marking:Marking),
                cont:Cont,
                conx:Conx),
            non_loc:NL),
   qstore:QStore,
   qretr:QRetr)
  **>
  (word,
   synsem:(loc:(cat:(head:(verb,
                           vform:fin,
                           aux:minus,
                           inv:Inv,
                           prd:Prd,
                           mod:Mod),
                        subcat:[NewSub|SubRest],
                        marking:Marking),
                cont:Cont,
                conx:Conx),
            non_loc:NL),
   qstore:QStore,
   qretr:QRetr)

  if pres_3s_act(Sub,NewSub)
  morphs
    (X,y) becomes (X,i,e,s),
    X becomes (X,s).

pres_3s_act((NP,@ np(_)),(NP,@ np((per:third,num:sing)),@ case(nom))) if
  !,true.
pres_3s_act(X,X) if
  true.

pres_non3s lex_rule
  (word,
   synsem:(loc:(cat:(head:(verb,
                           vform:bse,
                           aux:minus,
                           inv:Inv,
                           prd:Prd,
                           mod:Mod),
                        subcat:[Sub|SubRest],
                        marking:Marking),
                cont:Cont,
                conx:Conx),
            non_loc:NL),
   qstore:QStore,
```

```
   qretr:QRetr)
  **>
  (word,
   synsem:(loc:(cat:(head:(verb,
                           vform:fin,
                           aux:minus,
                           inv:Inv,
                           prd:Prd,
                           mod:Mod),
                     subcat:[NewSub|SubRest],
                     marking:Marking),
                cont:Cont,
                conx:Conx),
           non_loc:NL),
   qstore:QStore,
   qretr:QRetr)

  if pres_non3s_act(Sub,NewSub)
  morphs
    X becomes X.

pres_non3s_act((NP,@ np(_)),(NP,@ np((per:(first;second);num:plur)),
                           @ case(nom))) if
  true.

% exceptions: lexical entries

is --->                          % irregular auxiliary
  word,                          % ideally, this would be able to handle
  synsem:loc:(cat:(head:(verb, %  more than passive verb forms as predicates
                         mod:none,
                         vform:fin,
                         aux:plus),
                   subcat:[(NP,@ np((per:third,num:sing)),@ case(nom)),
                           (loc:(cat:(head:(vform:pas,
                                            aux:minus, % no "is been"
                                            prd:plus),
                                      subcat:[NP]),
                                 cont:Cont))],
                   marking:unmarked),
              cont:Cont,
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

% Passive Formation
% ------------------------------------------------------------------------
```

```
passive lex_rule
  (word,
   synsem:(loc:(cat:(head:(verb,
                           vform:bse,
                           aux:minus,
                           inv:Inv,
                           prd:Prd,
                           mod:Mod),
                     subcat:[_,
                             (loc:(cat:(head:(noun,
                                              prd:SubPrd,
                                              mod:SubMod), % ignore case
                                        subcat:[],
                                        marking:SubMarking),
                                   cont:SubCont,
                                   conx:SubConx),
                              non_loc:SubNL)|SubRest],
                     marking:Marking),
               cont:Cont,
               conx:Conx),
         non_loc:NL),
   qstore:QStore,
   qretr:QRetr)
  **>
  (word,
   synsem:(loc:(cat:(head:(verb,
                           vform:pas,
                           aux:minus,
                           inv:Inv,
                           prd:Prd,
                           mod:Mod),
                     subcat:[(loc:(cat:(head:(noun,
                                              prd:SubPrd,  % no case since
                                              mod:SubMod), %  not fin form
                                        subcat:[],
                                        marking:SubMarking),
                                   cont:SubCont,
                                   conx:SubConx),
                              non_loc:SubNL)|SubRest],
                     marking:Marking),
               cont:Cont,
               conx:Conx),
         non_loc:NL),
   qstore:QStore,
   qretr:QRetr)
```

```
  morphs
    give becomes given,
    see becomes seen,
    (X,y) becomes (X,ied),
    (X,e) becomes (X,ed),
    X becomes (X,ed).

% passive_agent LR goes here, pending analysis of PP's
% another passivization rule is also necessary for subcat<np,s> verbs,
%  because HPSG assigns accusative case in finite forms.  But should this one
%  be combined with it-extraposition, e.g. "I believe that J. is angry",
%  "*That J. is angry is believed", "It is believed that J. is angry"?
% another passive_"agent" rule is not necessary for subcat<s,np>, since the
%  passive rule can apply to the it-extraposed form

% It-Extraposition
% ------------------------------------------------------------------------

% regulars: lexical rule
it_extraposition lex_rule
  (word,
   synsem:(loc:(cat:(head:(Head,vform:bse),
                         subcat:Sub,
                         marking:Mark),
                 cont:Cont,
                 conx:Conx),
           non_loc:NL),
   qstore:QStore,
   qretr:QRetr)
  **>
  (word,
   synsem:(loc:(cat:(head:Head,
                         subcat:ExpSub,
                         marking:Mark),
                 cont:Cont,
                 conx:Conx),
           non_loc:NL),
   qstore:QStore,
   qretr:QRetr)

  if (append(Prev,[(S,@ s(_),loc:cat:marking:comp)|Rest],Sub),
      append(Rest,[S],NewRest),
      append(Prev,[(@ np(it))|NewRest],ExpSub))
  morphs
    X becomes X.

% exceptions: lexical entries
```

```
seems --->     % no sentential subjects allowed
  word,
  synsem:loc:(cat:(head:(verb,
                         mod:none,
                         vform:fin,
                         aux:minus,
                         inv:minus),
                   subcat:[(@ np(it)),
                           (@ s(SCont))],
                   marking:unmarked),
              cont:SCont,
              conx:backgr:e_set),
  (@ empty_non_loc),
  qstore:e_set.

% Subject Extraction
%-------------------------------------------------------------------------
subject_extraction lex_rule
  (word,
   synsem:(loc:(cat:(head:(Head,vform:bse),  % should look for reltvzr's
                     subcat:Sub,             % also, in order to generate
                     marking:Mark),          % SELR version of wh-
relativizer
                cont:Cont,                   % but that also relies on
                conx:Conx),                  % Raising Principle, which
           non_loc:(inherited:(slash:OldSlash, % cannot be implemented yet.
                               rel:Rel,
                               que:Que),
                    to_bind:ToBind)),
   qstore:QStore,
   qretr:QRetr)
  **>
  (word,
   synsem:(loc:(cat:(head:Head,
                     subcat:SESub,
                     marking:Mark),
                cont:Cont,
                conx:Conx),
           non_loc:(inherited:(slash:(elt:SlashLoc,elts:OldSlash),
                               rel:Rel,
                               que:Que),
                    to_bind:ToBind)),
   qstore:QStore,
   qretr:QRetr)

  if (append([Prev|Prevs],[(@ s(SCont),loc:cat:(head:SHead,
```

```
                                                  marking:unmarked),
                                        non_loc:(inherited:(rel:SRel,
                                                            que:SQue),
                                                 to_bind:SToBind))|Rest],Sub),
      append([Prev|Prevs],[(@ vp(SCont),loc:cat:(head:SHead,
                                                    subcat:[(loc:SlashLoc)],
                                                    marking:unmarked),
                                     non_loc:(inherited:(slash:e_set,
                                                         rel:SRel,
                                                         que:SQue),
                                              to_bind:SToBind))|Rest],SESub))
  morphs
    X becomes X.

% Grammar Rules
%
================================================================================

schema1 rule
(Mother,phrase,synsem:loc:cat:subcat:[])
===>
cat> (SubjDtr,non_word,synsem:SubjSyn),   % n.b. only one complement permitted
cat> (HeadDtr,phrase),
goal> (head_feature_principle(Mother,HeadDtr),
       inv_minus_principle(Mother),
       subcat_principle(Mother,HeadDtr,[SubjSyn]),
       marking_principle(Mother,HeadDtr),
       spec_principle(SubjDtr,HeadDtr),
       semantics_principle(Mother,HeadDtr,[SubjDtr]),
%      universal_trace_principle: obviated here by parochial
       parochial_trace_principle(SubjDtr),
%      subject_condition: not necessary - sch2,3,or word_promotion_1 did
       nonlocal_feature_principle(Mother,HeadDtr,[SubjDtr]),
       single_rel_constraint(Mother),
       clausal_rel_prohibition(Mother),
       relative_uniqueness_principle(Mother,[SubjDtr,HeadDtr]),
       conx_consistency_principle(Mother,[SubjDtr,HeadDtr]),
       deictic_cindices_principle(Mother,[SubjDtr,HeadDtr])).


schema2 rule
(Mother,phrase,synsem:loc:cat:subcat:[SubjSyn])
===>
cat> (HeadDtr,word,synsem:loc:cat:subcat:[SubjSyn|CompSyns]),
goal> synsems_to_non_words(CompSyns,Comps),
cats> (Comps,hd:FirstComp),
goal> (head_feature_principle(Mother,HeadDtr),
```

```
       inv_minus_principle(Mother),
       subcat_principle(Mother,HeadDtr,CompSyns),
       marking_principle(Mother,HeadDtr),
       spec_principle(FirstComp,HeadDtr),
       semantics_principle(Mother,HeadDtr,Comps),
       universal_trace_principle(Comps,HeadDtr),
%      parochial_trace_principle: subject not bound yet
       subject_condition(CompSyns,SubjSyn),
       nonlocal_feature_principle(Mother,HeadDtr,Comps),
       single_rel_constraint(Mother),
%      clausal_rel_prohibition: not necessary - mother has non-empty subcat
       relative_uniqueness_principle(Mother,[HeadDtr|Comps]),
       conx_consistency_principle(Mother,[HeadDtr|Comps]),
       deictic_cindices_principle(Mother,[HeadDtr|Comps])).

schema3 rule
(Mother,phrase,synsem:loc:cat:subcat:[])
===>
cat> (HeadDtr,word,synsem:(loc:cat:subcat:(SCompSyns,
                                           [SubjSyn|CompSyns]),
                           non_loc:to_bind:slash:e_set)),
goal> synsems_to_non_words(SCompSyns,SComps),
cats> (SComps,[Subj|Comps]),
goal> (head_feature_principle(Mother,HeadDtr),
       inv_plus_principle(Mother),
       subcat_principle(Mother,HeadDtr,SCompSyns),
       marking_principle(Mother,HeadDtr),
       spec_principle(Subj,HeadDtr),
       semantics_principle(Mother,HeadDtr,SComps),
       universal_trace_principle(Comps,HeadDtr), % UTP on FirstComp
       parochial_trace_principle(Subj),          %  obviated by parochial
       subject_condition(CompSyns,SubjSyn),
       nonlocal_feature_principle(Mother,HeadDtr,SComps),
       single_rel_constraint(Mother),
       clausal_rel_prohibition(Mother),
       relative_uniqueness_principle(Mother,[HeadDtr|SComps]),
       conx_consistency_principle(Mother,[HeadDtr|SComps]),
       deictic_cindices_principle(Mother,[HeadDtr|SComps])).

schema4 rule
(Mother,phrase)
===>
cat> (MarkDtr,phrase,synsem:loc:cat:(head:mark,
                                     subcat:[])),
cat> (HeadDtr,phrase,synsem:non_loc:to_bind:slash:e_set),
goal> (head_feature_principle(Mother,HeadDtr),
       inv_minus_principle(Mother),
```

```
       subcat_principle(Mother,HeadDtr,[]),   % no comp-dtrs
       marking_principle(Mother,MarkDtr),
       spec_principle(MarkDtr,HeadDtr),
       semantics_principle(Mother,HeadDtr,[MarkDtr]),
%      universal_trace_principle: not necessary - no comp-dtrs
%      parochial_trace_principle: not necessary - no comp-dtrs
%      subject_condition: not necessary - sch2,3 or word_promotion_1 will
       nonlocal_feature_principle(Mother,HeadDtr,[MarkDtr]),
       single_rel_constraint(Mother),
       clausal_rel_prohibition(Mother),
       relative_uniqueness_principle(Mother,[MarkDtr,HeadDtr]),
       conx_consistency_principle(Mother,[MarkDtr,HeadDtr]),
       deictic_cindices_principle(Mother,[MarkDtr,HeadDtr])).

schema5a rule
(Mother,phrase)
===>
cat> (AdjnDtr,phrase,synsem:loc:cat:(head:mod:Mod,
                                     subcat:[])),
cat> (HeadDtr,phrase,synsem:(Mod,
                             non_loc:to_bind:slash:e_set)),
goal> (head_feature_principle(Mother,HeadDtr),
       subcat_principle(Mother,HeadDtr,[]),   % no comp-dtrs
       marking_principle(Mother,HeadDtr),
%      spec_principle: not necessary - no comp-dtrs or marker-dtr
       semantics_principle(Mother,AdjnDtr,[HeadDtr]),
%      universal_trace_principle: not necessary - no comp-dtrs
%      parochial_trace_principle: not necessary - no comp-dtrs
%      subject_condition: not necessary - sch2,3 or word_promotion_1 will
       nonlocal_feature_principle(Mother,HeadDtr,[AdjnDtr]),
       single_rel_constraint(Mother),
       clausal_rel_prohibition(Mother),
       relative_uniqueness_principle(Mother,[AdjnDtr,HeadDtr]),
       conx_consistency_principle(Mother,[AdjnDtr,HeadDtr]),
       deictic_cindices_principle(Mother,[AdjnDtr,HeadDtr])).

schema5b rule
(Mother,phrase)
===>
cat> (HeadDtr,phrase,synsem:(Mod,
                             non_loc:to_bind:slash:e_set)),
cat> (AdjnDtr,phrase,synsem:loc:cat:(head:mod:Mod,
                                     subcat:[])),
goal> (head_feature_principle(Mother,HeadDtr),
       subcat_principle(Mother,HeadDtr,[]),   % no comp-dtrs
       marking_principle(Mother,HeadDtr),
%      spec_principle: not necessary - no comp-dtrs or marker-dtr
```

```
       semantics_principle(Mother,AdjnDtr,[HeadDtr]),
%      universal_trace_principle: not necessary - no comp-dtrs
%      parochial_trace_principle: not necessary - no comp-dtrs
%      subject_condition: not necessary - sch2,3 or word_promotion_1 will
       nonlocal_feature_principle(Mother,HeadDtr,[AdjnDtr]),
       single_rel_constraint(Mother),
       clausal_rel_prohibition(Mother),
       relative_uniqueness_principle(Mother,[AdjnDtr,HeadDtr]),
       conx_consistency_principle(Mother,[AdjnDtr,HeadDtr]),
       deictic_cindices_principle(Mother,[AdjnDtr,HeadDtr])).

schema6 rule
(Mother,phrase)
===>
cat> (FillDtr,phrase,synsem:(loc:FillLoc,
                             non_loc:inherited:slash:e_set)),
cat> (HeadDtr,phrase,synsem:(loc:cat:(head:(verb,
                                            vform:fin),
                                      subcat:[]),
                             non_loc:(inherited:slash:HeadSlashs,
                                      to_bind:slash:(elt:FillLoc,
                                                     elts:e_set)))),
goal> (set_member(FillLoc,HeadSlashs),
       head_feature_principle(Mother,HeadDtr),
       subcat_principle(Mother,HeadDtr,[]),     % no comp_dtrs
       marking_principle(Mother,HeadDtr),
%      spec_principle: not necessary- no comp-dtrs or marker-dtr
       semantics_principle(Mother,HeadDtr,[FillDtr]),
%      universal_trace_principle: not necessary - no comp-dtrs
%      parochial_trace_principle: not necessary - no comp-dtrs
%      subject_condition: not necessary - sch2,3 or word_promotion_1 will
       nonlocal_feature_principle(Mother,HeadDtr,[FillDtr]),
       single_rel_constraint(Mother),
       clausal_rel_prohibition(Mother),
       relative_uniqueness_principle(Mother,[FillDtr,HeadDtr]),
       conx_consistency_principle(Mother,[FillDtr,HeadDtr]),
       deictic_cindices_principle(Mother,[FillDtr,HeadDtr])).

word_promotion_0 rule
(phrase,synsem:Synsem,
        qstore:QStore,
        qretr:QRetr)
===>
cat> (word,synsem:(Synsem,loc:cat:subcat:[],
                          non_loc:to_bind:slash:e_set),
           qstore:QStore,
           qretr:QRetr).
```

```
word_promotion_1 rule
(phrase,synsem:Synsem,
        qstore:QStore,
        qretr:QRetr)
===>
cat> (word,synsem:(Synsem,loc:cat:subcat:[SubjSyn],
                          non_loc:to_bind:slash:e_set),
           qstore:QStore,
           qretr:QRetr),
goal> subject_condition([],SubjSyn).      % no other comps


% Macros
%
================================================================================

np(Ind) macro                                                          % p.
16
  loc:(cat:(head:noun,
              subcat:[]),
       cont:index:Ind).     % this one is NP "sub" i in the book, not NP:i

nbar(Cont) macro                                                        % p.
46
  loc:(cat:(head:noun,
              subcat:[(@ detp)]),
       cont:Cont).

case(Case) macro
  loc:cat:head:case:Case.

s(Proposition) macro
  loc:(cat:(head:verb,
            subcat:[]),
       cont:Proposition).

vp(Proposition) macro
  loc:(cat:(head:verb,
              subcat:[synsem]),
       cont:Proposition).

detp macro                                                             % p.
45
  loc:cat:(head:det,
             subcat:[]).
```

```
empty_non_loc macro
  synsem:non_loc:(inherited:(que:e_set,
                             rel:e_set,
                             slash:e_set),
                  to_bind:(que:e_set,
                           rel:e_set,
                           slash:e_set)).

empty_inher macro
  synsem:non_loc:inherited:(que:e_set,
                            rel:e_set,
                            slash:e_set).

% Principles
%
==========================================================================

% head_feature_principle(Mother,Head-Daughter)
%-----------------------------------------
head_feature_principle(synsem:loc:cat:head:X,synsem:loc:cat:head:X) if
  true.

% subcat_principle(Mother,Head-Daughter,Comp-Dtr-Synsems)
%-------------------------------------------------------------
subcat_principle((synsem:loc:cat:subcat:MSub),(synsem:loc:cat:subcat:HSub),
                 CompSyns) if
  append(MSub,CompSyns,HSub).

% marking_principle(Mother,Mark-Dtr)
%-----------------------------------------------------
% Mark-Dtr is marker-dtr, if any, o.w. head-dtr
marking_principle(synsem:loc:cat:marking:Mark,
                  synsem:loc:cat:marking:Mark) if
  true.

% spec_principle(Spec-Dtr,Head-Dtr)
%-------------------------------------------------------
%  Spec-Dtr is either mark-dtr or first comp-dtr
spec_principle((synsem:loc:cat:head:Head),synsem:HeadSynsem) if
  specp_act(Head,HeadSynsem).

specp_act(subst,_) if      % substantive head
  true.
specp_act(spec:X,X) if     % functional head
  true.

% semantics_principle(Mother,Semantic-Head,Other-Dtrs)
```

```
%-----------------------------------------------------
% Semantic-Head is adjunct-dtr, if any, o.w. head-dtr
semantics_principle((qstore:MQStore,
                     qretr:MRetr,
                     synsem:loc:cont:MCont),
                    (SHead,synsem:loc:cont:SCont),ODtrs) if
  qstores_of([SHead|ODtrs],e_set,DQStore),
  semp_act(SCont,MRetr,MQStore,MCont,DQStore).

semp_act((psoa,nucleus:Nucl,
               quants:SQuants),MRetr,MQStore,(nucleus:Nucl,
                                              quants:MQuants),DQStore) if
  !,set_sublist(MRetr,DQStore,MQStore),   % part (a)
  append(MRetr,SQuants,MQuants).          % part (b)
semp_act(Cont,[],QStore,Cont,QStore) if   % parts (a) and (b)
  true.

% universal_trace_principle(Comp-Dtrs,Head-Dtr)
%-----------------------------------------------
% The situation of the trace sort in the subsumption hierarchy, and the
%  type constraints on the participants of schemata guarantee that traces
%  will only appear as subcategorized elements.  The following ensures
%  that they will only appear as subcategorized by substantives.
universal_trace_principle([(trace)|_],HeadDtr) if
  !,utp_act(HeadDtr).
universal_trace_principle([_|Comps],HeadDtr) if
  universal_trace_principle(Comps,HeadDtr).
universal_trace_principle([],_) if
  true.

utp_act((synsem:loc:cat:head:subst)) if  % act predicate necessary for
  true.                                  % proper placement of cut above

% parochial_trace_principle(First-Comp-Dtr)
%-------------------------------------------
% strict subcategorization: exludes that-trace sentences
parochial_trace_principle(trace) if
  !,fail.
parochial_trace_principle(_) if
  true.

% subject_condition(Other-Comp-Dtr-Synsems,Subj-Dtr-Synsem)
%----------------------------------------------------------
subject_condition([],non_loc:inherited:slash:e_set) if
  true.
subject_condition([(non_loc:inherited:slash:ne_set)|_],_) if
  !,true.
```

```
subject_condition([(non_loc:inherited:slash:e_set)|CompSynRest],SubjSyn) if
  subject_condition(CompSynRest,SubjSyn).

% nonlocal_feature_principle(Mother,Head-Dtr,Other-Dtrs)
%-------------------------------------------------------
nonlocal_feature_principle((synsem:non_loc:inherited:(slash:MISlash,
                                                      que:MIQue,
                                                      rel:MIRel)),
                           (HeadDtr,synsem:non_loc:to_bind:(slash:HTSlash,
                                                            que:HTQue,
                                                            rel:HTRel)),
                           ODtrs) if
  islashes_of([HeadDtr|ODtrs],e_set,DISlash),
  iques_of([HeadDtr|ODtrs],e_set,DIQue),
  irels_of([HeadDtr|ODtrs],e_set,DIRel),
  set_diff(HTSlash,DISlash,MISlash),
  set_diff(HTQue,DIQue,MIQue),
  set_diff(HTRel,DIRel,MIRel).

% relative_uniqueness_principle(Mother,Dtrs)
%-------------------------------------------
% parochial(certain dialects): constrains the result of non-local feature
%  principle to prevent parasitic relatives
relative_uniqueness_principle(synsem:non_loc:inherited:rel:Rel,Dtrs) if
  rup_act(Rel,Dtrs).

rup_act(e_set,_) if
  true.
rup_act((elt:X,elts:Xs),Dtrs) if
  rup_elt(Dtrs,X),
  rup_act(Xs,Dtrs).

rup_elt([],_) if
  true.
rup_elt([(synsem:non_loc:inherited:rel:DRel)|DtrsRest],X) if
  set_member_eq(X,DRel),
  !,rup_elt_act(DtrsRest,X).     % belongs to one daughter
rup_elt([_|DtrsRest],X) if
  rup_elt(DtrsRest,X).

rup_elt_act([],_) if
  true.
rup_elt_act([(synsem:non_loc:inherited:rel:DRel)|DtrsRest],X) if
  (\+ set_member_eq(X,DRel)),
  rup_elt_act(DtrsRest,X).         % but no more than one
```

```
% conx_consistency_principle(Mother,Dtrs)
%-----------------------------------------
conx_consistency_principle((synsem:loc:conx:backgr:MBackgr),
                           Dtrs) if
  backgrs_of(Dtrs,e_set,MBackgr).


% deictic_cindices_principle(Mother,Dtrs)
%-----------------------------------------
deictic_cindices_principle((synsem:loc:conx:c_inds:MCinds),
                           Dtrs) if
  dcip_act(Dtrs,MCinds).

dcip_act([],_) if
  true.
dcip_act([(synsem:loc:conx:c_inds:DCinds)|DRest],DCinds) if
  dcip_act(DRest,DCinds).

% inv_minus_principle(Mother)
%---------------------------------
% parochial: if inv exists, it must be minus
inv_minus_principle(synsem:loc:cat:head:inv:Inv) if
  !,imp_act(Inv).              % inv is approp. and minus
inv_minus_principle(_) if
  true.                        %  or inapprop.

imp_act(minus) if
  true.

% inv_plus_principle(Mother)
%---------------------------------
% parochial: if inv exists, it must be plus
inv_plus_principle(synsem:loc:cat:head:inv:Inv) if
  !,ipp_act(Inv).             % either inv is approp. or causes failure
inv_plus_principle(_) if
  true.                        %  and succeeds here

ipp_act(plus) if
  true.

% single_rel_constraint(Sign)
%-------------------------------
% parochial: Rel set can't have more than one element
% enforced on words and trace as type constraint; enforced on phrases as
%  procedural attachment to rules
single_rel_constraint(synsem:non_loc:inherited:rel:e_set) if
  true.
```

```
single_rel_constraint(synsem:non_loc:inherited:rel:elts:e_set) if
  true.

% clausal_rel_prohibition(Sign)
%---------------------------------
% parochial: Sentences must have empty Rel set
clausal_rel_prohibition((synsem:non_loc:inherited:rel:e_set)) if  % empty Rel
  true.
clausal_rel_prohibition((synsem:(non_loc:inherited:rel:ne_set,
                                 loc:cat:head:(func
                                               ;adj
                                               ;noun
                                               ;prep
                                               ;reltvzr)))) if
  true.                                                  % not a verb
projn.
clausal_rel_prohibition((synsem:(non_loc:inherited:rel:ne_set,
                                 loc:cat:(head:verb,
                                          subcat:ne_list)))) if
  true.                                                  % not a sentence


% Utilities
%
==========================================================================

% union(?set1,+set2,?union)
union(e_set,Xs,Xs) if
  true.
union((elt:X,elts:Xs),Ys,Zs) if
  set_member_eq(X,Ys),
  !,union(Xs,Ys,Zs).
union((elt:X,elts:Xs),Ys,(elt:X,elts:Zs)) if
  set_select(X,Ys,YsRest),
  union(Xs,YsRest,Zs).
union((elt:X,elts:Xs),Ys,(elt:X,elts:Zs)) if
  union(Xs,Ys,Zs).

set_member(X,(elt:X)) if
  true.
set_member(X,(elts:S)) if
  set_member(X,S).

set_member_eq(X,(elt:Y)) if
  (X =@ Y).
set_member_eq(X,(elts:S)) if
  set_member_eq(X,S).
```

```
set_select(X,(elt:X,elts:Xs),Xs) if
  true.
set_select(Member,(elt:X,elts:Xs),(elt:X,elts:Rest)) if
  set_select(Member,Xs,Rest).

% set_select_eq(+member,+set,?rest)
set_select_eq(X,(elt:Y,elts:Xs),Xs) if
  (X =@ Y).
set_select_eq(Member,(elt:X,elts:Xs),(elt:X,elts:Rest)) if
  set_select_eq(Member,Xs,Rest).

append([],Xs,Xs) if
  true.
append([H|T1],L2,[H|T2]) if
  append(T1,L2,T2).

% selectors
%-------------
% in the X_of predicates, testing first for e_set means that if we don't
%  specify that feature, then, by default, it is the empty set

backgrs_of([],MBackgr,MBackgr) if
  true.
backgrs_of([(synsem:loc:conx:backgr:e_set)|DRest],Accum,MBackgr) if
  backgrs_of(DRest,Accum,MBackgr),
  !.
backgrs_of([(synsem:loc:conx:backgr:DBackgr)|DRest],Accum,MBackgr) if
  union(Accum,DBackgr,NewAccum),
  backgrs_of(DRest,NewAccum,MBackgr).

qstores_of([],QStores,QStores) if
  true.
qstores_of([(qstore:e_set)|Dtrs],Accum,QStores) if
  qstores_of(Dtrs,Accum,QStores),
  !.
qstores_of([(qstore:DQStore)|Dtrs],Accum,QStores) if
  union(Accum,DQStore,NewAccum),
  qstores_of(Dtrs,NewAccum,QStores).

islashes_of([],ISlash,ISlash) if
  true.
islashes_of([(synsem:non_loc:inherited:slash:e_set)|Dtrs],Accum,ISlash) if
  islashes_of(Dtrs,Accum,ISlash),
  !.
islashes_of([(synsem:non_loc:inherited:slash:DISlash)|Dtrs],Accum,ISlash) if
  union(Accum,DISlash,NewAccum),
```

```
  islashes_of(Dtrs,NewAccum,ISlash).

iques_of([],IQue,IQue) if
  true.
iques_of([(synsem:non_loc:inherited:que:e_set)|Dtrs],Accum,IQue) if
  iques_of(Dtrs,Accum,IQue),
  !.
iques_of([(synsem:non_loc:inherited:que:DIQue)|Dtrs],Accum,IQue) if
  union(Accum,DIQue,NewAccum),
  iques_of(Dtrs,NewAccum,IQue).

irels_of([],IRel,IRel) if
  true.
irels_of([(synsem:non_loc:inherited:rel:e_set)|Dtrs],Accum,IRel) if
  irels_of(Dtrs,Accum,IRel),
  !.
irels_of([(synsem:non_loc:inherited:rel:DIRel)|Dtrs],Accum,IRel) if
  union(Accum,DIRel,NewAccum),
  irels_of(Dtrs,NewAccum,IRel).

set_sublist([],Set,Set) if
  true.
set_sublist([X|Subs],Set,RestSet) if
  set_select(X,Set,Rest),
  set_sublist(Subs,Rest,RestSet).

set_diff(e_set,Set,Set) if   %first arg should be instantiated, so cut
  !,true.                    % in case it isn't, and it should also be subset
                             % of the second
set_diff((elt:X,elts:Xs),Set,Diff) if
  set_select_eq(X,Set,Rest),
  set_diff(Xs,Rest,Diff).

synsems_to_non_words([],[]) if   % cut is very important - nothing has
  !,true.                        %  guaranteed that inputs are sufficiently
                                 %  instantiated
synsems_to_non_words([Syn|Synsems],[(non_word,synsem:Syn)|Signs]) if
  synsems_to_non_words(Synsems,Signs).
```