```
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%
%
%
% File:             ch3_kokota.dtr
%
% Purpose:          the kokota noun phrase
%
% Author:           Dunstan Brown (with thanks to Roger Evans)
%
%                   26 November 2011
%
% Email:            d.brown@surrey.ac.uk
%
% Address:          University of Surrey, Guildford, GU2 7XH, UK
%
% Documentation:   'Network Morphology', Brown & Hippisley 2012
%
% Related files:
%
% Version:          0.07
%
%
%
% % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
%


%
%
%                   SYNTAX
%
%
%
% NP1 is an NP with a noun head only
%
% NP2 is an NP with a noun following the noun head
%
% NP3 is an NP with an adjective following the noun head
%
% NP4 is an NP with a verb following the noun head
%
% <syn isedge> tracks edges - by default everything is an edge, but
%
% gets marked as non_edge if it acquires a right sibling
%

NP1:
    <> == UNDEF
    <syn> == undefined
    <syn form> == "<syn head form>"
    <syn isedge> == edge
    <syn head> == HEAD
    <syn head lex> == "<n1>".

NP2:
    <> == NP1
    <syn form> == NP1 "<syn comp form>"
    <syn isedge head> == not_edge
    <syn comp> == COMP
    <syn comp lex> == "<n2>".

NP3:
    <> == NP2
    <syn comp lex> == "<adj>".

NP4:
    <> == NP2
    <syn comp lex> == "<v>".


%
%
%                  HEAD AND COMPLEMENT FEATURES
%
%
%
%
%
% HEAD defines the form of the head of the phrase, by collecting up
%
% all the head features and passing them to the head lexeme.
%
% It adds a dem marker if the head is at the edge
%
% It also require that the head be a noun.
%
% COMP defines complements similarly. In addition, it disallows head-only
%
% nouns from occurring in complement position
%


HEAD:
    <> == UNDEF
    <syn head form> == "<syn head lex word "<syn head feat>" >"
    <syn head feat> == <syn "<syn head lex cat>" >
    <syn n> == <syn "<syn isedge head>" >
    <syn edge> == <syn not_edge> "<syn dem>"
    <syn not_edge> == "<syn poss>".
```

```
COMP:
    <> == UNDEF
    <syn comp form> == "<syn comp lex word "<syn comp feat>" >"
    <syn comp feat> == <syn "<syn isedge comp>" >
    <syn edge> == <syn "<syn comp lex cat>" >
    <syn n head> == UNDEF
    <syn> == "<syn dem>".


%
%
%                   MORPHOLOGY
%
% We are only required to define the morphology of words.
%
% The demonstrative clitic can attach to any word class and is therefore
%
% given at the MOR_WORD node.
%
%
%
% As we note on page 119 in section 3.1.1, this node can be merged into
%
% the lexemic hierarchy under Node Elimination, because the inheritance
%
% structure mirrors that of the lexemic hierarchy. No morphological
%
% hierarchy is therefore actually required.
%
%
%
% As discussed in section 3.1.1, neither head nor edge are referred to in
%
% the equations defining the morphology.
%



MOR_WORD:
    <mor> ==
    <mor dem pl not_visible> ==  -ro
    <mor dem sg not_visible> ==  -no
    <mor dem sg within_reach> == -ine
    <mor poss first sg> == -gu "<mor>"
    <mor poss first pl> == -mai "<mor>".

%
%
```

```
%                  LEXEME HIERARCHY
%
%
%
% Fairly self-explanatory. The morphology of words defaults to MOR_WORD.
%
% A word consists of a stem and some morphology. A stem is by default a
%
% root.
%


WORD:
    <mor> == MOR_WORD
    <word> == "<stem>" "<mor>"
    <stem> == "<root>".

NOUN:
    <> == WORD
    <cat> == n.

% IND_POSS is marked as a head-only noun - it cannot occur in complement
%
% position. In addition, it can only be realised if <syn poss> is defined.
%
% The path <word undefined> at the node IND_POSS ensures that a possessor
%
% indexing host must have a specification for morphosyntactic features
%
% associated with possession, person and number.
%

IND_POSS:
    <> == NOUN
    <cat> == n head
    <word undefined> == UNDEF.

VERB:
    <> == WORD
    <cat> == v.

ADJ:
    <> == WORD
    <cat> == adj.

%
%
%                  EXAMPLE LEXICAL ENTRIES
%
%
```

```
%
%
%


Nene:
    <> == NOUN
    <root> == nene
    <gloss> == leg.

Mane:
    <> == NOUN
    <root> == mane
    <gloss> == man.

Ge:
    <> == IND_POSS
    <root> == ɣe
    <gloss> == consumable_possession.

Vave:
    <> == NOUN
    <root> == vave
    <gloss> == in_law_relative.

Tove:
    <> == ADJ
    <root> == tove
    <gloss> == old.

Dou:
    <> == VERB
    <root> == dou
    <gloss> == be_big.

Kame:
   <> == NOUN
   <root> == kame
   <gloss> == hand.

%
%
%                  EXAMPLE NOUN PHRASES
%
%
%
%
%
% These nodes are used to create queries. Note the last one, which tries
%
```

```
% to insert the verb 'dou' into a head position. This will fail, because
%
% the head of the np must be a noun.
%
%
%
% The path <expected> states what the correct forms are. It is not referred
%
% to anywhere in the theory.
%

EXAMPLEA:
    <> == NP1
    <n1> == "Nene:<>"
    <syn poss> == poss first sg
    <expected> == nene -gu.

EXAMPLEB:
    <> == NP2
    <n1> == "Ge:<>"
    <n2> == "Nene:<>"
    <syn poss> == poss first sg
    <expected> == ɣe -gu nene.

EXAMPLEC:
    <> == NP4
    <n1> == "Mane:<>"
    <v> == "Dou:<>"
    <syn dem> == dem pl not_visible
    <expected> == mane dou -ro.

EXAMPLED:
    <> == NP2
    <n1> == "Mane:<>"
    <n2> == "Vave:<>"
    <syn dem> == dem pl not_visible
    <expected> == mane vave -ro.

EXAMPLEE:
    <> == NP1
    <n1> == "Nene:<>"
    <syn poss> == poss first sg
    <syn dem> == dem pl not_visible
    <expected> == nene -gu -ro.

EXAMPLEF:
    <> == NP1
    <n1> == "Nene:<>"
    <syn poss> == poss first pl
    <expected> == nene -mai.
```

```
EXAMPLEG:
    <> == NP2
    <n1> == "Ge:<>"
    <n2> == "Nene:<>"
    <syn poss> == poss first pl
    <expected> == ɣe -mai nene.

EXAMPLEH:
    <> == NP3
    <n1> == "Mane:<>"
    <adj> == "Tove:<>"
    <syn dem> == dem sg not_visible
    <expected> == mane tove -no.

EXAMPLEI:
    <> == NP4
    <n1> == "Mane:<>"
    <v> == "Dou:<>"
    <syn dem> == dem sg not_visible
    <expected> == mane dou -no.

EXAMPLEJ:
    <> == NP2
    <n1> == "Mane:<>"
    <n2> == "Vave:<>"
    <syn dem> == dem sg not_visible
    <expected> == mane vave -no.

EXAMPLEK:
    <> == NP1
    <n1> == "Nene:<>"
    <syn poss> == poss first pl
    <syn dem> == dem sg not_visible
    <expected> ==nene -mai -no.

EXAMPLEL:
    <> == NP2
    <n1> == "Ge:<>"
    <n2> == "Nene:<>"
    <syn poss> == poss first pl
    <syn dem> == dem sg not_visible
    <expected> == ɣe -mai nene -no.

EXAMPLEM:
    <> == NP1
    <n1> == "Nene:<>"
    <expected> == nene.

% Examples in the book, with nodes labelled according to the example
```

```
%
% numbering in chapter 3.
%

EXAMPLE15:
    <> == NP3
    <n1> == "Mane:<>"
    <adj> == "Tove:<>"
    <syn dem> == dem pl not_visible
    <expected> == mane tove -ro.

EXAMPLE16:
    <> == NP1
    <n1> == "Kame:<>"
    <syn poss> == poss first sg
    <syn dem> == dem sg within_reach
    <expected> == kame -gu -ine.

EXAMPLE17:
    <> == NP2
    <n1> == "Ge:<>"
    <n2> == "Nene:<>"
    <syn poss> == poss first sg
    <syn dem> == dem pl not_visible
    <expected> == ɣe -gu nene -ro.

% SHOULD_FAIL1 fails, because it tries to insert a verb as NP head.
%

SHOULD_FAIL1:
    <> == NP1
    <n1> == "Dou:<>"
    <expected> == **fail**.


% SHOULD_FAIL2 fails, because the indirect possession host is not inserted
%
% as the head of the NP.
%

SHOULD_FAIL2:
    <> == NP2
    <n1> == "Nene:<>"
    <n2> == "Ge:<>"
    <expected> == **fail**.

% SHOULD_FAIL3 fails for the same reason as SHOULD_FAIL2, because the
%
% indirect possession host is not inserted as the head of the NP.
%
```

```
% Here we also provide the morphosyntactic features for possession, but the
%
% it still fails.
%


SHOULD_FAIL3:
    <> == NP2
    <n1> == "Nene:<>"
    <n2> == "Ge:<>"
    <syn poss> == poss first sg
    <syn dem> == dem sg
    <expected> == **fail**.

SHOULD_FAIL4:
    <> == NP2
    <n1> == "Ge:<>"
    <n2> == "Nene:<>"
    <expected> == **fail**.

# uses UNDEF.

#show
    <expected>
    <syn form>
    <syn poss>
    <syn dem>.

#hide
NP1 NP2 NP3 NP4 MOR_WORD WORD NOUN VERB ADJ Nene Mane Ge
Kame Vave Tove Dou HEAD COMP UNDEFINED IND_POSS.
```