Appendix I. An Implementation on Unicorn3 Parsing System[1]

/*[2] Appendix I is an implementation of the proposed grammar on the unicorn3 parsing system.[3] Interpretation of Semantic Representations are:

Given a content consisting of [REL$^{[\text{CASE } \alpha]}$ φ] and [ARG ψ], the variable that the lambda binds a free occurrence of in ψ in computing $\mathbf{\{x^{[CASE\ \alpha]} \mid \psi\} \in \varphi}$ (equivalently, $\mathbf{\varphi(\lambda x^{[CASE\ \alpha]}[\psi])}$) is the one that is case-registered as the same as the feature REL whose value is φ. If the case value is QUAN, as in the content consisting of [REL$^{\text{QUAN}}$ φ] and [ARG ψ] then the computation is $\mathbf{\{X \mid (X, \{x^{QUAN} \mid \psi\}) \in \varphi\}}$.

I have tested the implementation, and the implementation works, as explained in the main text. */

% 1. Rules
% 1.1. Sentence Rule
/* This phrase rule defines an object of the sort of [MAJ *v*] and [VFORM *finite*] and [COMPS *end*] as a sentence. */
x0 → x1:
<x0 maj>=s[4]
<x0 head>=<x1 head>
<x0 comps>=<x1 comps>
<x0 hd_arg_st>=<x1 hd_arg_st>
<x0 content>=<x1 content>
<x1 maj>=v
<x1 head vform>=finite
<x1 comps>=end.

% 1.2. Phrase Rules
/* There are three phrase rules in the grammar. All the Phrase rules share the scheme of *x0 → x1 x2:* */

define Head_Feature_Principle_of_Phrase:
<x0 maj>=<x2 maj>
<x0 head>=<x2 head>
<x0 hd_arg_st>=<x2 hd_arg_st>.

define Non_Head_Daughter_Comps_Saturated:

---

[1] Unicorn3 was developed at the University of Illinois at Urbana-Champaign.

[2] The parser ignores all the letters that are to the right of % that are on the same line as %, and all the letters between /* and */.

[3] Unicorn3 cannot have a typed-feature theory written.

[4] Unicorn3 cannot have features typed. MAJ is not a head feature in this implementation. If it were, then the sentence rule would be inconsistent.

/* The COMPS of the non-head daughter are “saturated”. */
<x1 comps>=end.

define Content_of_Argument_Predicate_Phrase:
/* If the non-head daughter works as an argument, then the content of the entire phrase that the non-head daughter adjoins to structure-shares with that of the head daughter. */
<x0 content>=<x2 content>.

define Comp_Head_Phrase:
<x2 comps first>=[]
<x1>=<x2 comps first>
<x0 comps>=<x2 comps rest>.

%1.2.1. COMP-HEAD PHRASE
/* The complement-head-phrase rule consists of the descriptions that refer to the definitions above. */
x0 → x1 x2:
Head_Feature_Principle_of_Phrase
Non_Head_Daughter_Comps_Saturated
Comp_Head_Phrase
Content_of_Argument_Predicate_Phrase.

define Adjunct_Head_Phrase:
<x0 comps>=<x2 comps>
<x1 head mod>=[]
<x2>=<x1 head mod>.

%1.2.2. ARGUMENT/ADJUNCT-HEAD PHRASE
/* The non-head daughter serves as an argument of the head daughter semantically, and yet, is an adjunct to the head daughter. */
x0 → x1 x2:
Head_Feature_Principle_of_Phrase
Non_Head_Daughter_Comps_Saturated
Adjunct_Head_Phrase
Content_of_Argument_Predicate_Phrase.

define Content_of_Nonargument_Predicate_Phrase:
<x0 content>=<x1 content>.

%1.2.3. ADJUNCT-HEAD PHRASE
/* The adjunct-head phrase is more specifically a non-argument-and-adjunct head phrase. That is, the non-head daughter does not serve as an argument of the constituent semantically, and is an adjunct to the constituent. */
x0 → x1 x2:
Head_Feature_Principle_of_Phrase
Non_Head_Daughter_Comps_Saturated
Adjunct_Head_Phrase
Content_of_Nonargument_Predicate_Phrase.

% 2. LEXICON

```
% 2.1. CASE MORPHEME
define Case:
<maj>=k
<comps first maj>=n
<comps rest>=end.

define Nom:
<head mod maj>=v
<head mod head vform>=finite
<head kform>=nom.

  % 2.1.1. Typical Nominative Morpheme
  Word ga:
  Case
  Nom
  <head mod hd_arg_st rel_nom>=<comps first content>.

  % 2.1.2. “Topic-like” Nominative Morpheme
  Word ga:
  Case
  Nom
  <content arg>=<head mod content>
  <content rel_nom_or_acc_or_gen>=<comps first content>.

define Acc:
<head mod maj>=v
<head kform>=acc.

  % 2.1.3. Accusative Morpheme
  Word o:
  Case
  Acc
  <head mod hd_arg_st rel_acc>=<comps first content>.

define Gen:
<head mod maj>=n
<head kform>={nom acc gen}.

  % 2.1.4. Genitive Morpheme
  Word no:
  Case
  Gen
  <content arg>=<head mod content>
  <content rel_nom_or_acc_or_gen >=<comps first content>.

% 2.2. VERB
define finite:
<head mod>=no
<head vform>=finite.
```

define Verb:
<maj>=v
<comps>=end
<content>=<hd_arg_st>..

% 2.2.1. INTRANSITIVE VERB
Word neru:
Finite
Verb
<hd_arg_st arg>=sleep_xn_. % sleep'(x^{NOM})

% 2.2.2. TRANSITIVE VERB
Word taberu:
Finite
Verb
<hd_arg_st arg>=eat_ya__xn_. % (eat'(y^{ACC}))(x^{NOM})

% 2.3. NOUN
define Noun:
<maj>=n
<comps>=end
<head mod>=no
<content>=<hd_arg_st>
<content rel_q>=some_. % some' = {(X, Y) | X ∩ Y ≠ ∅}.

% 2.3.1. John
Word zyon:
Noun
<content arg>=John_xq_. % John'(x^{QUAN})

% 2.3.2. someone's child
Word kodomo:
Noun
<content arg>=child_xq_and_r_xq__yg_. % child'(x^{QUAN}) & R(x^{QUAN})(y^{GEN})

% 2.3.3. someone's cake
Word keeki:
Noun
<content arg>=cake_xq_and_r_xq__yg_. % cake'(x^{QUAN}) & R(x^{QUAN})(y^{GEN})