```
noun-lex := lex-item &
  [ SYNSEM.LOCAL.CAT [ HEAD noun,
                       VAL [ SPR < synsem >,
                             COMPS < >,
                             SUBJ < >,
                             SPEC < > ]]].
```

This type specifies that the HEAD value (part of speech) is *noun*, and what to do with the valence features. There is a specifier requirement, but no subject, complements or SPEC (we'll get back to this one).

i. Reload the grammar (LKB/Load/Reload grammar), and fix any errors that the LKB reports.
ii. Look at the type definition for *noun-lex* within the lkb (LKB/View/Type definition). This should show what you just added (in a slightly different format).
iii. Now look at the expanded type for *noun-lex* (LKB/View/Expanded type). All of that additional information is inherited from the supertypes you specified.
iv. Constrain the HEAD value of the specifier requirement (if any) to be *det*. Use the expanded type view to help you figure out what the path is that you need to use for that feature. (Hint: The elements of valence lists are objects of type *synsem*. Also, you can join an identity tag and a feature constraint with &: #spr & [...].)
v. Open the file lexicon.tdl and replace the dummy entry (foo) with a noun instance, inheriting from *noun-lex*, e.g.,:

```
penguin := noun-lex &
  [ STEM < "penguin" >].
```

Use fully inflected forms for now. We'll add inflectional rules later. This lexical entry is also missing semantics. We'll get to that later.
vi. Open the file roots.tdl and add the following root condition:

```
root := lex-item.
```

(This says that a lexeme or a word can serve as a stand-alone utterance.)
vii. Try to parse a sentence consisting of just your noun (e.g., "penguin").
viii. Now add the types *intrans-verb-lex* and *trans-verb-lex* for intransitive verbs and transitive verbs (inheriting from *lex-item*), analogously for what we did for nouns above. Constrain their SUBJ lists to have a *synsem* with the HEAD value *noun*. Give them the appropriate value of COMPS. Let the SPR be empty. SPEC should also be empty (it's used for determiners and degree specifiers of adjectives, etc).
ix. Add lexical entries (in lexicon.tdl) for your transitive and intransitive verbs, again sticking with fully inflected forms and not worrying about agreement for now.
x. Add particles/determiners as necessary. For determiners, define a subtype of *lex-item*, *det-lex*, with the head value *det*, and let the SPEC list have a *synsem* with the head value *noun*. Let the other valence lists be empty.
xi. Reload the grammar, and make sure you can parse each of the individual lexical items.

(c) Add node labels

The LKB reads the file labels.tdl to figure out what labels to stick on nodes (to abbreviate those big feature structures). You'll find it easier to work with the grammar if you add some. Label names look like this:

```
n-label := label &
  [ SYNSEM.LOCAL.CAT [ HEAD noun,
                       VAL.SPR < synsem > ],
    LABEL-NAME "N"].
```

If those constraints are compatible with the node, the LKB will use it. If not, it will read the next label, etc. (NB: This means that the order in which the node labels are defined is important.)

Add labels for N, NP, V, VP, S and D. The conventional use of these symbols as abbreviations is as follows:

i. N: HEAD *noun*, non-empty SPR list
ii. NP: HEAD *noun*, empty SPR list
iii. V: HEAD *verb*, non-empty COMPS list
iv. VP: HEAD *verb*, empty COMPS list, *synsem* on the SUBJ list
v. D: HEAD *det*,
vi. S: HEAD *verb*, empty COMPS list, *anti-synsem* on the SUBJ list

(d) Add language-specific rules to your grammar

i. Determine whether the subject precedes or follows the verb in your language (or both). Introduce a head-subj-phrase which inherits from *basic-head-subj-phrase* and either *head-final* or *head-initial*, as appropriate.
ii. Now add a rule instance (or two, if necessary) to the file rules.tdl, like this:

```
head-subj := head-subj-phrase.
```

iii. Analogously, add types and instances for head-complement phrases (to combine verbs with their objects, here) and head-specifier phrases (to combine nouns with determiners, here).
iv. Open the file roots.tdl, comment the root condition we added above out, and uncomment the other one that's there.
v. Try parsing a sentence with the intransitive verb. If it doesn't parse, debug.
vi. Now try parsing a sentence with the transitive verb. Are there more parses than you would expect? If so, constrain the COMPS value of the head daughter of the head subject rule to rule some out.
vii. Try parsing a sentence with a transitive verb again, and examine the parse chart (LKB/Parse/Show parse chart). Are there more edges (phrases) than you would expect? If so, consider whether they are really legit, and if not, how you might rule them out.

(e) Add inflection

i. Change *noun-lex*, *intrans-verb-lex*, and *trans-verb-lex* to *noun-lxm*, *intrans-verb-lxm*, and *trans-verb-lxm*. Let *noun-lxm* inherit from the matrix type *lexeme*. Introduce a type *verb-lxm* as a subtype of *lexeme* and stipulate the information that your verb lexeme types have in common. Let *intrans-verb-lxm* and *trans-verb-lxm* inherit from the new type and delete the redundant information.
ii. Change *det-lex* to *det-word*, and let it inherit from the matrix type *word*.
iii. Update your lexicon and change the inflected forms of your nouns and verbs to base forms.
iv. Introduce word types for nouns and verbs. Let them inherit from *word* and give them appropriate HEAD values.
v. Introduce types for singular definite nouns and present tense verbs without worrying about agreement information. These subtypes should inherit from the relevant word types plus the right lexeme-to-word types indicating whether they add inflection or not. A word type which adds inflection will inherit from the matrix type *infl-ltow-rule* (for example English "barks"), while a word type which doesn't add any inflection should inherit from the type *const-ltow-rule* ("bark"). The word type for third singular present verbs in an English grammar may look like this:

```
3sing-verb-word := verb-word & infl-ltow-rule.
```

vi. Introduce inflectional rules in the "irules.tdl" file. An inflectional rule which adds inflection looks like this:

```
3sing-pres-v_infl_rule :=
  %suffix (* s)
  3sing-pres-verb-word.
```

Remember that if a rule does not add any inflection, you simply unify the STEM value of the daughter with the STEM value of the mother:

```
non3sing-pres-v_infl_rule := non3sing-pres-verb-word &
  [ STEM #1,
    ARGS < [ STEM #1 ] > ].
```

vii. Change the *root* type in "roots.tdl" so that only words can parse (see above) and check that your inflectional rules work. If you have ensured that, change it back and test sentences.

5. Test your grammar

   (a) Make a list of sentences that your grammar should parse (made up of the words you've added today). (At least 4.)

   (b) Make a list of sentences that your grammar shouldn't parse (i.e., ungrammatical sentences) with the words you added today. (At least 12.)

   (c) Put the lists into a file called test.items.

   (d) Put a * before each of the ungrammatical sentences.

   (e) Use the LKB batch test facility to check whether your grammar behaves appropriately, and debug as necessary.

      i. LKB/Parse/Batch parse
      ii. Specify matrix/test.items as the input file.
      iii. Specify matrix/test.out as the output file.
      iv. When the LKB is done, examine test.out. The number after the sentence indicates how many parses it got. (A 0 after a sentence indicates that it didn't parse.)