Håkan Burden
burden@chalmers.se

Compositional Logical Semantics in Prolog
I have implemented my solution using SWI-prolog.

Task 1: VP Structures
Intransitive verbs have the same semantic interpretations as nouns:

\[ iv(\lambda(X, \text{arrive}(X))) \rightarrow \text{[arrives]} \]
\[ vp(IV) \rightarrow iv(IV) \]

This is also true for predicatives with an important difference. The copula verb 'is' has no semantic interpretation of its own, instead it acts like a glue and sticks the predicative attribute following it onto the subject.

\[ vp(PRED) \rightarrow \text{cop}, \text{pred(PRED)} \]
\[ \text{cop} \rightarrow \text{[is]} \]

Verbs with prepositional compliments are treated as transitive verbs made up of two words:

\[ tv(\lambda(P, \lambda(X, \text{leaves_for}(X, Y)))) \rightarrow \text{[leaves_for]} \]

Task 2: Quantified NPs
This was the trickiest task, according to me. For implicature I have the following semantic expression:

\[ \text{det}(\lambda(X, \lambda(Y, \forall(Z, \text{implies}(X@Z, Y@Z)))) \rightarrow \text{[every]} \]

It is derived from the fact that implicature is a binary relation over an individual Z. For both occurrences of Z in the relation we need to introduce a quality X and Y. X is introduced by the following noun and should therefore be combined with the first occurrence of Z. Y comes from the combining the resulting noun phrase with a verb phrase.

The same goes for the existential quantifier but here the binary relation is a conjunction of two qualities.

\[ \text{det}(\lambda(X, \lambda(Y, \exists(Z, \text{and}(X@Z, Y@Z)))) \rightarrow \text{[some]} \]

All we need now is a rule combining a determiner with a noun into a noun phrase:

\[ np(\text{DNN}) \rightarrow \text{det(D), noun(N)} \]
Task 3: Complex NPs

Adjectives are on the form:

\[ \text{adj}(\lambda(X, \lambda(Y, \text{and}(\text{large}(Y), X@Y)))) \rightarrow \text{[large]} \].

The conjunction \text{and} is used to give the \( Y \) under discussion both the quality \text{large} and the quality \( X \), which is introduced by the noun it modifies. So even if predicative attributes can be seen as adjectives they have a different syntactic role which gives them a different semantic interpretation.

I tried two solutions for modifying nouns with adjectives. The first solution introduced the rules:

\[
\begin{align*}
\text{ap}(\text{ADJ}) & \rightarrow \text{adj}(\text{ADJ}). \\
\text{ap}(\text{ADJ@AP}) & \rightarrow \text{adj}(\text{ADJ}), \text{ap}(\text{AP}). \\
\text{noun}(\text{AP@N}) & \rightarrow \text{ap}(\text{AP}), \text{noun}(\text{N}).
\end{align*}
\]

This solution gave trees on form seen to the left in figure 1. After some consideration I decided to use the solution shown in the right tree in figure 1. This meant only adding one rule:

\[ \text{noun}(\text{A@N}) \rightarrow \text{adj}(\text{A}), \text{noun}(\text{N}). \]

The reason for choosing the last solution was that I preferred the recursive noun modification over the lop-sided ap-solution.

---

**Figure 1: Adjective modification of nouns.**

```
noun
  ap
  noun
    adj
    ap
      noun
        adj
        noun
          large
          adj
            city
          large
            adj
              city
            scandinavian
```

---

**Example run:**

Apart from the grammar rules the file \texttt{grammar.pl} also contains the predicate \texttt{interpret/2} which takes a sentence and gives the semantic representation.

\[
\text{?- interpret([sas,flies, to, every, large, scandinavian, city], X).}
\]

\[
X = \text{forall(\(_G2095, \text{implies}(\text{and}(\text{large}(\(_G2095), \text{and}(\text{scandinavian}(\(_G2095), \text{city}(\(_G2095)), \text{flies_to}(\text{sas, \(_G2095))))\))}
\]

Yes