– Übung 8 –

The Farmer, the Wolf, the Goat and the Cabbage Return

1 Setting Up

This Übung will require you to write a grammar with at least one left recursive rule, so ordinary DCGs are not going to work. A suitable version of an Earley parser is in my home directory, called `uebung8.pl`. Copy this file into your working directory.

Note that this version of the parser expects rules to be written in a slightly DCG-like format. So, instead of writing

```
rule( s(...), [np(...),vp(...)] ).
```

this parser is expecting grammar rules to be written

```
s(...) ---> [ np(...), vp(...) ].
```

This format lends itself a little better to rules that are split across several lines, which in turn makes augmented phrase structure rules easier to read. The format of the lexicon (the `word/2` predicate) is unchanged.

2 The Problems

One nice thing about the world of the Farmer's Dilemma is that all of the inhabitants of that world are unique: there is only one farmer, one wolf, one boat, one river, etc. This means that we can talk about one particular state of that world with only definite NPs, and avoid the problems introduced by quantified NPs. The only exceptions are the banks of the river: I recommend just using `north` and `south` to refer to the two banks.

1 The Model First, build a model, that is, a particular state of affairs in which the four individuals `farmer`, `wolf`, `goat` and `cabbage` as well as `boat` are all given definite locations. For example, you might include facts like `in(farmer,boat)` and `on(boat,river)` to indicate that the boat is in the middle of the river and the farmer is currently in the boat.

2 Simple Sentences Now write DCG rules for S, NP, VP and PP that generate (among others) the following sentences.

The farmer is in the boat.
The boat is on the river.
The boat is in the north.

Your grammar should assign all these sentences logical forms which can be run directly as Prolog goals and checked against the model you built in Step 1.

Parse some test sentences and evaluate their logical forms against the model. You can do this by issuing goals of the following form.

```
?- s(LF,[the,farmer,is,in,the,boat],[]), call(LF).
```

or just

```
?- s(LF,[the,farmer,is,in,the,boat],[]), LF.
```

(As usual, you might prefer to implement a `parse/2` relation instead of calling `s/3` directly.)

3 Conjunction and Disjunction Now implement rules for conjunctions and disjunctions of clauses. So you should now be able to recognize and analyze sentences like

The goat is in the south and the cabbage is in the north.
The wolf is in the boat or the wolf is in the north.

Remember that the logical forms you assign should be Prolog goals that can be called directly and will succeed or fail depending on particular details of the model from Step 1.

4 Negation Implement negation, assigning it Prolog negation-as-failure for its semantics. So you should now be able to handle sentences like

The goat is not in the boat.

5 And Finally Of course if you want to keep the syntax simple, then all the NPs need to be of the same semantic type. And so if the language has any quantified NPs in it, *all* NPs need to be raised to that type, including proper names and definite descriptions, which otherwise might be taken simply to denote individuals.

Accordingly, a sentence like *the goat is in the boat* should really be assigned a logical form like

```
the(A,goat(A),the(B,boat(B),in(A,B)))
```

If we want to stick to our goal of using callable Prolog goals as our logical forms, then we must make `the/3` a callable Prolog goal. That is, you must provide a definition of predicate `the(Variable,Restrictor,Nucleus)` so that it succeeds or fails whenever the sentence it represents is true or false, respectively.

Optional: Note that *the* carries with it a presupposition of uniqueness: there is something funny about an NP like *the goat* in a world in which there are two goats. Can you make your semantics encode that uniqueness presupposition? That is, can you make it so that if you have a model with two goats, both in the boat, the sentence *the goat is in the boat* comes out false?

If you don't worry about uniqueness, then it is possible to implement `the/3` *without any calls to* `setof/3` *or anything nasty like that.*

You should now alter the NP and S rules in your grammar, plus the lexical entry for *the*, so as to assign them this new denotation. This should not affect the truth values of any of the sentences you could handle before.