MA410 Prolog Practical 6 - Definite Clause Grammars and NLP

1 Definite Clause Grammars

Definite Clause Grammar (or DCG for short) provides a syntax for providing more readable grammar parsing rules, without including linked difference lists. DCG Notation uses the '`-->`' operator.

(a) Download file `lab6dcg.pl`.
You should see the following code there.

```
sentence --> np, vp.
np --> det, adj, noun.
np --> adj, noun.
np --> det, noun.
vp --> verb, np.
vp --> verb, prep, np.
```

`np`, `vp` are shorthand for 'noun phrase' and 'verb phrase' respectively.

(b) Using the example '`noun --> [mouse].`',
define the following:

- `noun`(s): `dog, flies, fruit, banana`
- `verb`(s): `chases, eats, flies, like`
- `det`(erminers): `a, the`
- `prep`(ositions): `like`
- `adj`(ectives): `nice, hungry, fruit`

(c) Try '`?- listing(sentence).`' and '`?- listing(np).`'
Note the generated linked difference lists.

(d) Use the `trace` facility to follow the processing in

```
?- sentence([fruit, flies, like, a, banana],[]).
```

Note again the difference list processing.

(e) Find all possible legal sentences.

2 Grammatical Structure

In order to retrieve the structure of a sentence and to check grammar rules we must provide the structure within our definitions of the predicate.

(a) Download and edit file `lab6gra.pl`. The grammatical structure is held within functors `sentence, nphrase, vphrase, noun` & `verb`. The predicate names used are 2-letter combinations (not a requirement but a convention for ease of use) formed by:

se = **s**entenc**e**, **ne** = **n**oun phras**e**,
ve = **v**erb phras**e**, **dr** = **d**etermine**r**,
nn = **n**ou**n**, **vb** = **v**er**b**.

(b) Try the code:

```
?- se(GS, [the,man,eats,the,apples],[]).
```

```
?- se(GS, Snt, []).
```

Here `GS` represents the grammatical structure for the sentence.

3 Word Agreement

We need to go a step further in order to provide agreement between words, e.g. we say "the dog eats" for singular but "the dogs eat" for plural.

(a) Download file `lab6wagr.pl`. Note the extra parameter list in predicate definitions, e.g. `[Num, AI]`.

(b) Try the code:

```
?- se(GS, [the,man,eat,some,apple],[]).
?- se(GS, [the,men,eat,some,apples],[]).
?- se(GS, [all,men,eat,apples],[]).
?- se(GS, Snt, []).
```

(c) Add to the file:

- `det`(erminers): `all`, `this`, `an`
- `noun`(s): `woman`, `women`, `water`
- `verb` structures: `drink`, `drinks`
- `pronoun`(s): `i`, `you`, `he`, `she`,... etc.

(d) Change the code as follows:

- Full grammatical correctness of `an` and `a`.
- Create a property `Person` (= `first`, `second` or `third`) and change definitions accordingly.
- Fully conjugate verbs, e.g. I eat, you eat, he/she/it eat<u>s</u>, we/ye/they eat.
- Include the verb '`to be`' in the present tense.

4 Prompting for sentences

(a) Include the file `sent2lst.pl` in `lab6wagr.pl`.

(b) Type `?- read_in(X).` and see what happens.

(c) Use the predicate `read_in(X)` in conjunction with predicates above to allow for ordinary sentence input, e.g. define

```
rsent(GS,X) :- read_in(X), se(GS,X,[]).
```

(d) Try out some examples.

Task: (i) "The dog drinks water" is correct but "The dog eats apple" is not. The reason is that water is an uncountable noun. Change the code above to distinguish between countable and uncountable nouns. (ii) Change the definitions to allow pronouns.

Include an example in each case. E-mail the code with subject heading "Lab DCGs".