Course on Artificial Intelligence and Intelligent Systems

Natural language analysis with DCG and HYPROLOG:

Comments on the recommended reading, examples and exercises

Henning Christiansen
Roskilde University, Computer Science Dept.
 Version 12 oct 2007

This note gives some examples on the recommended literature, and gives some examples of grammars and exercises. Chapters 7 and 8, minus 7.2.3 and 8.1.3 of the text [1], which gives an introduction to Definite Clause Grammars, plus the user's guide to HYPROLOG [5] are compulsory reading. Articles [2, 3] are meant as background reading (not compulsary); the first one describes HYPROLOG, its implementation and background in more detail, whereas the second gives a theoretical account on this approach to discourse analysis. We refer below to example files that are available on the course ecampus area.

1 Definite clause grammars

Most Prolog systems include the popular Definite clause grammar notation, colloquially DCG, which makes it very easy to implement simple language analyzers. DCGs have been used for fragments of natural language as well as programming languages, mostly for analysis and translation, but it is interesting to notice that that they can be used also for *generation* of text. However, in the course we are mostly interested in analysis.

You should be aware that DCGs only take care of part of what normally is expected for a language processing system. There is no straightforward way to have a text file analyzed; if that is what you want, you need to do a lot of programming yourself. A DCG works on *Prolog lists* which in principle can contain arbitrary elements, but in typical applications constant symbols (in Prolog terminology: atoms). For example, if you have written a

little DCG for simple sentences, having a nonterminal called `sentence`, the following example query may show how you can activate analysis.

```
?- phrase(sentence,[logic,programming,is,fun]).
yes
```

(Here we used the SICStus convention using the `phrase` predicate; some systems may do it in another way.)

Most textbooks on Prolog has a chapter on DCG, and in this course we use chapters 7 and 8 of [1] which is available online. It is suggested that you skip section 7.2.3 and 8.1.3.

Comments on the text

Many introductions to DCG, including chapter 7 of [1], tend to explain DCGs in terms of their implementation which may not be that smart from a pedagogical point of view. A better way (in the present writers view) will be to explain DCGs in terms of example grammars and then leaving the implementation principles to an appendix.

In [1], the authors start explaining that you can do language analysis by means of the `append` predicate, then they explain you that this is not a good idea. Next they explain s technique called difference lists and conclude that this is a better way to implement language analysis (difference lists *are* smart, but it may be difficult for the novice to see this). Finally, they show you the grammar notation. At the KIIS course, we are interested in the applications of DCGs and very little in the implementation. However, [1] is fairly easy to read, so it is suggested that you read the chapters 7 and 8 from the beginning. In fact, the only things you really need to care about the concerning the implementation of DCGs are the following:

- A top-down, left-to-right parsing strategy is used.
- Backtracking is applied in order to find the rules that should be applied.
- In case of an ambiguous grammar, you'll get the alternative analyses by backtracking (e.g., typing semicolon).
- If your grammar rules are left-recursive, it will most likely run into infinite loops.

2 Discourse analysis as abduction

We refer here to two recent research papers [3, 2] which may be a bit difficult to read, and you'll most like not understand everything.

A programming language called HYPROLOG is introduced in [2] which is based on CHR and Prolog. It includes abduction implemented in the way we have seen already in the course, and it includes also a related notion of assumptions which are useful for language analysis; small examples and references are given in the paper. Most importantly, the paper indicates how abduction and assumptions can be applied together with the DCG notation. HYPROLOG is available on the web [4] and needs to be loaded into SICStus Prolog.

The second paper, [3], entitled "Meaning in Context" gives a theoretical account on this way of using abduction for language analysis. It may be a bit tough to read if you are not used to this sort of theoretical papers, but try anyhow — it's not that complicated after all. This paper refers to a language called A^2LP, which is a forerunner of HYPROLOG (basically the same!).

3 A first DCG

The following program is copied from [1] and shows a DCG without extra arguments. Available on file `dcg1`.

```
np  --> det,n.

vp  --> v,np.
vp  --> v.

det --> [the].
det --> [a].

n   --> [woman].
n   --> [man].

v   --> [shoots].
```

4 Adding extra arguments

The following program is copied from [1] and shows a DCG with extra arguments in order do describe how `he-him` and `she-her` should be applied correctly. Available on file `dcg2`.

```
s     --> np(subject),vp.

np(_) --> det,n.
np(X) --> pro(X).
```

```
vp    --> v,np(object).
vp    --> v.

det   --> [the].
det   --> [a].

n     --> [woman].
n     --> [man].

pro(subject) --> [he].
pro(subject) --> [she].
pro(object)  --> [him].
pro(object)  --> [her].

v --> [shoots].
```

Exercise 1

The grammar should be extended so that it includes plural-singular distinction. The grammar above has only singular, so to make it interesting, add the following:

- nouns `men`, `women`,
- pronouns `they, them`,
- verb `shoot`.

You will need to add an extra argument to some of the nonterminals to hold the number which should be one of `sing`, `plu` (for singular and plural).

Write the grammar in such a way that it only accepts grammatical correct sentences such as "the women shoot the men", "she shoots the man", but not "she shoot the man".

Implement it on the computer and test it.

Exercise 2

Extend the grammar so that it will accept sentences such as the following.

He shoots her with a water pistol.

A woman shoots him with a gun.

NB: It is suggested that you write the grammar so that you can accept any `np` after `with`. So, for example, the following sentence will be syntactically legal although semantically problematic.

The water pistol shoots the gun with a woman.

5 Adding extra conditions

The following little grammar describes the command language for a little robot that can walk along a straight line. The grammar uses the curly-bracket notation in order to assign a "meaning" to a command sequence, which is the final position of the robot (assuming that it starts at position `0`).

Curly brackets are useful for conditions that cannot be expressed in an easy way using unification of arguments only.

The grammar is available at file `trip`. Notice that there are two nonterminals called `trip`, but they differ in the number of arguments.

```
trip(To) --> trip(0,To).

trip(Here,Here) --> [].

trip(From,To) -->
   step(HowMuch),
   {NewFrom is From + HowMuch},
   trip(NewFrom,To).


step(1) --> [forward].

step(-1) --> [back].

step(0) --> [think].
```

The following shows a query and answer for this grammar.

```
?- phrase(trip(N), [forward,forward,think,back,think,forward,forward]).
N = 3
```

Exercise 3

Calculate by hand how the Prolog system got to this answer in the example above.

Exercise 4

The robot in the example above is now modified so that it walks in a 2D space. Extend the command language in the previous example with commands `up` and `down`, and (using two arguments, one for x and one for y coordinates) modify the grammar so that it determines the final position in the 2D space.

6 On possible worlds and abduction

Here we consider, in an informal way, the Meaning-In-Context model described in [3].

The next exercise fit fine for a small programming project, but it is to time-consuming to be solved as such in the class. You may solve it using Prolog and CHR or HYPROLOG.

The idea is to discuss possible solutions and to understand the idea of possible worlds and interpretation as abduction (from an intuitive point of view).

We give a simplistic exercise following, which involves implementation by few lines of code.

Exercise 5

Let us consider again the shooting language and to make it simple, we assume to begin with that all sentences refer to named persons (no pronouns for now), and that a person is uniquely identified by that name. Let us assume that the following persons can be talked about, *Lucky Luke*, *Joe Dalton*, *Jack Dalton*, *William Dalton*, *Averell Dalton*, *Calamity Jane*, and `Ma Dalton`.

We consider discourses about shooting incidents, reported in the order in which they have occurred. The set of worlds that we consider, corresponds to all worlds in which we can imagine for Lucky Luke comics. However, for simplicity it as assumed that an incident of shooting is fatal for the one being shot at.

Notice that this example includes a notion of time which may make the terminology a bit difficult. So when we here refer to a "possible world", it is not only the state-of-affairs at a specific moment, but it includes a history of all past and future events. (Probably the people who invented the term "possible worlds" were thinking of static worlds, but the concept is also applied for dynamic words as well.)

It may be useful to assume a clock so that first sentence describes an incident at time 0, the next sentence an incident at time 1, etc.

Question 5.1

What sort of context facts (i.e., "abducibles") will be relevant for analysis of discourses in the indicated language?
NB: You may give a preliminary answer and go back and revise it when you continue with the following questions.

Question 5.2

Describe the contexts (i.e., set of context facts, i.e., set of abducibles) of possible worlds after each of the following discourses.

- The empty discourse, i.e., nothing has been said. (Leading hint: You cannot assume that someone is alive due to lack of reports of the opposite.)
- *Lucky Luke shoots Jack Dalton.*
- *Lucky Luke shoots Jack Dalton. Joe Dalton shoots Ma Dalton.*
- *Lucky Luke shoots Joe Dalton. Joe Dalton shoots Ma Dalton. Joe Dalton shoots Lucky Luke.*

Question 5.3

Write down the integrity constraints you used (explicitly or implicitly) when answering the previous questions. (It is OK to write them down in everyday language).

Question 5.4

Pronoun resolution is a difficult matter which is problematic not only in computer systems for language processing but also for humans. We recall that pronouns are words such as "he", "him", "her", etc., and *pronoun resolution* is a matter of finding out which individual a specific occurrence of, say, "he" refers to.

You need not consider any smart way of implementing this for this question, and it can be answered by reasoning in an informal way. We will assume only one thing, namely that an individual needs to be mentioned at least once in a discourse before he-or-she can be referred to later in the discourse.

You task is here similar to the previous question, except that you should make the analysis of discourses in an extended language which includes pronouns.

Describe the contexts (i.e., set of context facts, i.e., set of abducibles) of possible worlds after each of the following discourses. Explain also how you resolved the pronouns.

- *Lucky Luke shoots Jack Dalton. He shoots Joe Dalton.*
- *Lucky Luke shoots Jack Dalton. Joe Dalton shoots Calamity Jane. He shoots Ma Dalton.*
- *Lucky Luke shoots Joe Dalton. Averell Dalton shoots Lucky Luke. He shoots Calamity Jane.*

- *Joe Dalton shoots Lucky Luke. He shoots him.*

7 Assumptions for pronoun resolution

Let us recall the the following definition for assumptions taken from [2].

`+h(a)`	Assert linear assumption for subsequent proof steps. Linear means "can be used once".
`*h(a)`	Assert intuitionistic assumption for subsequent proof steps. Intuitionistic means "can be used any number of times".
`-h(X)`	Expectation: consume/apply existing int. assumption.
`=+h(a)`, `=*h(X)`, `=-h(X)`	Timeless versions of the above, meaning that order of assertion of assumptions and their application or consumption can be arbitrary.

The exercise below needs only the standard versions (non-timeless) but we will show an example from [2] using the timeless ones. Timeless assumptions make it possible to refer forwards into the discourse, and are useful for many things, including ellipsis and coordination.

The discourse *"Peter likes and Mary hates Martha"* contains two coordinating sentences in the sense that the first incomplete one takes its object from the second one. This can be described by having an incomplete sentence put forward a timeless expectation that may be satisfied by a later assumption produced by a complete sentence; the following two grammar rules are sufficient.

```
sentence(s(A,V,B)) --> np(A), verb(V), np(B), {=*obj(B)}.
sentence(s(A,V,B)) --> np(A), verb(V), [and],   {=-obj(B)}.
```

In this form, we used an argument to represent the "meaning" of a phrase and not abducibles as we have discussed. The following form would also works, provided `s/3` is consider to be an abducible.

```
sentence --> np(A), verb(V), np(B), {=*obj(B), s(A,V,B)}.
sentence --> np(A), verb(V), [and],   {=-obj(B), s(A,V,B)}.
```

Such assumptions are available in HYPROLOG; notice that the actual assumptions used, needs to be declared as described in the HYPROLOG manual [5].

Exercise 6

Consider the following grammar for a highly simplified version of the shooting language. It is given as a HYPROLOG source file, `dcg3`; comments in the file explain how it should be started.

Now we use a new nonterminal called `discourse` which is defined directly as a sequence of sentences `ss`. In case of a pronoun it does not do anything smart at all, but simply recognizes the referred individual as `anonymous`. Sentence meanings are emitted as abducibles `event/3`.

```
% ?- [hyprolog].
% ?- hyprolog(dcg3).  % perhaps  ?- hyprolog('dcg3.txt').

abducibles

   event/3. % example: event(shooting,maDalton,luckyLuke)

discourse --> ss.
ss    --> [].    % ss stands for sentenceS
ss    --> s, ss.

s    --> np(_,Who), [shoots], np(_,Whom),
         {event(shooting,Who,Whom)}.

np(Gender,Who)   --> pro(Gender,Who).
np(Gender,Who)   --> name(Gender,Who).

name(masc,luckyLuke)      --> [luckyLuke].
name(masc,joeDalton)      --> [joeDalton].
name(masc,jackDalton)     --> [jackDalton].
name(masc,williamDalton)  --> [williamDalton].
name(masc,averellDalton)  --> [averellDalton].
name(fem,calamityJane)    --> [calamityJane].
name(fem,maDalton)        --> [maDalton].

pro(masc,anonymous) -->  [he].
pro(fem,anonymous) -->   [she].
pro(masc,anonymous)  --> [him].
pro(fem,anonymous)  -->  [her].
```

The following example shows queries and answers corresponding to language analyses with this grammar.

```
?-  phrase(discourse, [luckyLuke,shoots,calamityJane]).
event(shooting,luckyLuke,calamityJane) ?

?- phrase(discourse, [luckyLuke,shoots,calamityJane, he,shoots,maDalton]).
event(shooting,luckyLuke,calamityJane),
event(shooting,anonymous,maDalton) ? ;
```

Intuitively, the `anonymous` in the last `event` should have been `luckyLuke`. Your task is to extend this grammar with assumptions so that we get the expected answer; as above, it is only possible to refer to individuals already mentioned in the discourse. You can use assumptions directly within curly-bracket notation if you declare them in the way shown in the HYPROLOG manual.

For the solution to this exercise, it is not expected that you prevent dead character from shooting.

8 Programming project (not expected to be done by all students!)

Here we describe a possible programming project which should not be solved in the class but can be selected for the written assignment (details on what is expected for the solution will be described elsewhere).

The task is to write a more complete version of a grammar for the shooting language which performs a discourse analysis with integrity constraints. The grammar should include the following aspects:

- distinction of pronouns `he-him` and similar depending on subject or object position,
- abducibles for `event` that include a time stamp,[1]
- abducibles for `dead` and `alive` which include also time stamps, so that you can write ...
- integrity constraints that prevent dead people from shooting, and
- pronoun resolution for singular pronouns.

It is suggested that you avoid language constructions such as *"... with a water pistol"*.

The aspects listed below may or may not be included, but you are strongly advised to wait adding these things until you have finished and documented a solution that includes the above. It is not required that you include any of these additional aspects, but here they are given for inspiration They are given in order of (expected) difficulty.

- Add plural pronouns `they` and `them` and adjust the grammar so that pronoun resolution works in a reasonable way. (NB: You will likely not produce a solution that always produces the intuitively correct results.)

[1] Ask your teacher for a hint in how to implement the time.

- Extend with `nps` of the form `[a,man]` and `[a,woman]` (using rules with `det` as shown in sections 1 and 2. You may resolve those in the same way as the pronouns, but you may also try to experiment with ways to refer forward.

- Encode in some ways, a preference so that `he` will refer to the most recently mentioned male character who is alive, and similarly for other references. It may be that `[a,man]` refers to forward to the closest relevant male person, and `[the,man]` backwards in a similar way.[2]

References

[1] Patrick Blackburn, Johan Bos, and Kristina Striegnitz. Learn Prolog Now!, 2005. Online document, `http://www.coli.uni-saarland.de/~kris/learn-prolog-now/`; also available as pdf, `http://www.coli.uni-saarland.de/~kris/learn-prolog-now/html/prolog-notes.pdf`

[2] H. Christiansen and V. Dahl. HYPROLOG: a new approach to logic programming with assumptions and abduction. In Maurizio Gabbrielli and Gopal Gupta, editors, *Proceedings of Twenty First International Conference on Logic Programming (ICLP 2005)*, Lecture Notes in Computer Science, 2005. To appear.

[3] H. Christiansen and V. Dahl. Meaning in Context. In Anind Dey, Boicho Kokinov, David Leake, and Roy Turner, editors, *Proceedings of Fifth International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-05)*, volume 3554 of *Lecture Notes in Artificial Intelligence*, pages 97–111, 2005.

[4] Henning Christiansen. HYPROLOG: A Logic Programming Language with Assumptions and Abduction. `http://www.ruc.dk/~henning/hyprolog/`, 2005.

[5] Henning Christiansen. User's guide to the HYPROLOG system: A logic programming language with assumptions and abduction. `http://www.ruc.dk/~henning/hyprolog/HyprologUsersGuide.html`, 2005.

[2] You should ask your teacher for a hint or two if you try to wrestle with preferences in CHR!