Course on Artificial Intelligence and Intelligent Systems

Natural language analysis with assumptions in Hyprolog:

Introduction, examples and exercises

Henning Christiansen
Roskilde University, Computer Science Dept.
 Version 9 oct 2008

This note continues the previous on natural language analysis in Prolog and CHR. Here we will make use of the Hyprolog system [4], which adds a layer of syntactic sugar upon Prolog and CHR with facilities for abductive reasoning and for another notion of *assumptions* which is explained below. You are referred to the Hyprolog User's Guide [5] for all details about the system. Notice that Hyprolog integrates smoothly with the DCG notation as well as all other facilities of Prolog and CHR. Hyprolog is developed for SICStus Prolog 3, but the new release contains also versions for SICStus Prolog 4 and SWI Prolog.

Articles [1, 2] 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 Getting started

Soursce code for download is available at the Hyprolog website [4], and the Hyprolog User's Guide [5] explains how to install and start the system.

A source file to Hyprolog looks very much like a Prolog+CHR source text, but specialized declarations and facilities are made available.

2 Abduction in Hyprolog

The implementation of abduction Hyprolog is based on the principles we have seen in the course. If you want to define an abducible predicate, asy `a/2` for use in a Prolog program (e.g., a DCG), you can define it in Hyprolog in the following way.

```
abducibles a/2.
```

Now you can use this in the same as as if you had declared (with SICStus 4 and SWI syntax):

```
:- chr_constraint a/2.
```

However, the `abducibles` declaration introduces also additional facilities that are described in the User's Guide, and that we do not need here.

If you want to see what Hyprolog does for your declarations of abducibles (and other Hyprolog specifics), you may add the following command as the first thing in your source file:

```
show_internal_rules.
```

3 Assumptions in Hyprolog

(This section quoted from [1])

Assumptive logic programs [7] are logic programs augmented with a) linear, intuitionistic and timeless implications scoped over the current continuation, and b) implicit multiple accumulators, useful in particular to make the input and output strings invisible when a program describes a grammar (in which case we talk of Assumption Grammars [8]). More precisely, we use the kind of linear implications called *affine* implications, in which assumptions can be consumed at most once, rather than exactly once as in linear logic. Although intuitively easy to grasp and to use, the formal semantics of assumptions is relatively complicated, basically proof theoretic and based on linear logic [7, 8, 9]. Here we use a more recent and homogeneous syntax for assumptions introduced in [3]; we do not consider accumulators, and we note that Assumption Grammars can be obtained by applying the operators below within a DCG.

`+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.
`expections_satisfied`	Tests whether all expectations have been met by an assumption; should be applied as the last thing in a query.

A sequential expectation cannot be met by timeless assumption and vice versa, even when they carry same name. In [8], a query cannot succeed with a state which contains an unsatisfied expectation; for simplicity (and to comply with our implementation), this is not enforced in HYPROLOG but can be tested explicitly using the primitive called `expections_satisfied`. Assumption grammars have been used for natural language problems such as free word order, anaphora, coordination, and for knowledge based systems and internet applications. Assumptive logic programs are useful, among other things, for simulation of producer-consumer and resource allocation systems, as illustrated in [6].

We show only the sequential versions below; assumptions work well together with abduction for discourse analysis as we will consider in an exercise below.

4 An example of a language analysis problem which uses sequential assumptions

We consider situations with three agents `a`, `b`, and `c` Agent `a` may drop objects on the floor and `b` and `c` may take them up. We use assumptions to describe which objects are on the floor and which objects that `b` and `c` may eventually have. The full Hyprolog program that defines a grammar for a suitable language and with assumptions is as follows; it is available at the course with the name `simpleAsump`.

```
assumptions on_floor/1, has/2.

story --> [] ; s, ['.'], story.

s -->  [a, drops], object(O), {+on_floor(O)}.

s -->  anotherAgent(A), [pics,up,something], {-on_floor(X), +has(A,X)}.
```

```
object(keys) --> [keys].
object(money) --> [money].

anotherAgent(b) --> [b].
anotherAgent(c) --> [c].
```

Exercise

Load the grammar into the Hyprolog system and test some queries, including the following ones.

```
?- phrase(story, [a,drops,keys,'.',b,pics,up,something,'.',
                  a,drops,money,'.',c,pics,up,something,'.']).

?- phrase(story, [a,drops,keys,'.',a,drops,money,'.',
                  b,pics,up,something,'.',c,pics,up,something,'.']).
```

Exercise

Insert the following command in the beginning od the file and load it into Hyprolog again.

```
show_internal_rules.
```

Try to explain, as best as you can, the CHR that are generated automatically in order to interpret the operators for assumptions and expectations.[1] You will probably see facilities that you have not met before, but in that case try to guess what they may be doing.

Exercise

Now you need to write your own grammar with assumptions, however, some bits and pieces are provided for you here, which you may copy from here and enter into your own source text. We consider now a language with the same three agents `a`, `b`, `c`, but now any of the three may both drop and pick up objects. The following is an example of the sort of queries that the new grammar should be able to interpret.

```
?- phrase(story, [a,drops,keys,'.',b,pics,up,something,'.',
                  a,drops,money,'.',c,pics,up,something,'.',
                  b,drops,food,'.',a,pics,up,something,'.']).
```

We assume the assumption declarations from the example above,

```
assumptions on_floor/1, has/2.
```

[1] Notice that the Hyprolog versions for different platforms generates slightly different rules.

The grammar rules for sentences will be of the following, where you have to figure out what to put into the curly brackets according to the detailed explanation given below.

```
s -->  agent(A), [drops], object(O), {....}.
s -->  agent(A), [pics,up,something], {....}.
```

We require for this grammar that an agent can only drop an object if it actually has that object. This will require that the grammar sets up an initial state of assumptions concerning who initially has which objects.

```
story  --> {initialize}, story1.
story1 --> [] ; s, ['.'], story1.

initialize:- *has(a,money), +has(a,keys), +has(b,food).
```

Question

Explain what these grammar rules and the `initialize` predicate means. Can you explain especially what the start on the first assumptions does (as opposed to the plusses).?

Question

Fill in the remaining grammar rule and complete the twe rules for sentences (`s`) above.

Question (only if you have time)

Change the grammar such that we do not need the initialization part, but instead the language contains a kind of sentences by means of which you can state that an agent has a given object without picking them up.

4.1 Exercise

Combine firstly the grammar of section 5 in the note from last week, with the solution to exercise 1 in the same note which introduced pronouns, and kept tract of number and case. In other words, the grammar should be able to analyze correctly, a sentences such as *"Peter sees her"*.

Secondly, you should now extend the grammar with assumptions in order to provide an interpretation of pronouns. This means that the sentence above (when preceeded by one or more sentences sentence that refer to Jane), may be interpreted as a set of abducibles that includes `see(peter,jane)`. The ideas is that you should use an assumption of form

```
+acting(jane,fem)
```

to represent the information that Jane has been mentioned in a sentence and this may be candidate for being referred to by "she" or "her". Notice that yoy will need to introduce information concerning gender (such as `fem` for feminine and *masc* for masculine).

4.2 Topic for a possible written assignment

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. Assume furthermore that an incident of shooting is always fatal for the one being shot at (which means the one cannot shoot after being shot).

This implied notion of time can be implemented by a kind of "clock" that follows the discourse such that the first sentence describes an incident at time 0, the next sentence an incident at time 1, etc.

```
story --> story(0).  % a whole story is a story that starts at time 0
story(Time)  --> []  ; s(Time), {TimePlus1 is Time+1}, story(TimePlus1).
```

When you resolve pronouns, you should make sure that you only in the end select indiviuals that are actually alive at the time the take part in an incident. You should not modify the assumption-expectation mechanism (!!), but you may express the restrictions in terms of integrity constraints (i.e., CHR rules about abducibles). It may or may not be useful to introduce extra abducibles.

References

[1] 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.

[2] 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.

[3] Henning Christiansen. CHR Grammars. *Int'l Journal on Theory and Practice of Logic Programming*, 2005. To appear.

[4] Henning Christiansen. HYPROLOG: A Logic Programming Language with Assumptions and Abduction. `http://www.ruc.dk/~henning/hyprolog/`, 2005-2008.

[5] Henning Christiansen. User's guide to the HYPROLOG system: A logic programming language with assumptions and abduction. `http://akira.ruc.dk/~henning/hyprolog/HyprologUsersGuide.html/`, 2005-2008.

[6] Henning Christiansen and Veronica Dahl. Assumptions and abduction in Prolog. In Elvira Albert, Michael Hanus, Petra Hofstedt, and Peter Van Roy, editors, *3rd International Workshop on Multiparadigm Constraint Programming Languages, MultiCPL'04; At the 20th International Conference on Logic Programming, ICLP'04 Saint-Malo, France, 6-10 September, 2004*, pages 87–101, 2004.

[7] Verónica Dahl and Paul Tarau. Assumptive logic programming. In *Argentine Symposium on Artificial Intelligence (ASAI), Cordoba, Argentina*, 2004.

[8] Verónica Dahl, Paul Tarau, and Renwei Li. Assumption grammars for processing natural language. In *ICLP*, pages 256–270, 1997.

[9] Paul Tarau, Verónica Dahl, and Andrew Fall. Backtrackable state with linear affine implication and assumption grammars. In Joxan Jaffar and Roland H. C. Yap, editors, *ASIAN*, volume 1179 of *Lecture Notes in Computer Science*, pages 53–63. Springer, 1996.