

Re: predicate in a predicate

Source: <http://sci.tech-archive.net/Archive/sci.logic/2007-12/msg00334.html>

- *From:* Keenlearner <yingun@xxxxxxxxxx>
 - *Date:* Sat, 8 Dec 2007 06:12:04 -0800 (PST)
-

On Dec 8, 9:10 pm, Jan Burse <janbu...@xxxxxxxxxxxx> wrote:

Keenlearner schrieb:

Thank you Herbzet,
Now I know how to differentiate function and predicate.

Thanks to Jan Burse, that is a nice resource. I will allow time to read at it.

I found a Prolog natural language program from
www.mtome.com/Publications/PNLA/prolog-digital.pdf
I have extend a little bit on it, for sentence input of "Every student is on a table" it will generate the FOL

Let's say
A "Universal quantifier"
E "Existential quantifier"
and "conjunction"
or "disjunction"
=> "implication"

$A x (\text{student}(x) \Rightarrow E y (\text{table}(y) \text{ and } \text{on}(x, y)))$

is this correct ?

Re: predicate in a predicate

Whether it's correct or not, as you can see there is two conclusions or literals.

Prolog is somehow from Horn Clauses, they can have at most one positive literals.

In this case, how can I convert the FOL that have two positive literals into Prolog ? Thank you.

Pure Prolog alone doesn't help. But with negation as failure you can pretty well model a lot but not all of FOL logic quite straight forward.

With negation as failure you can go beyond horn clauses. And usually one does not talk about clauses anymore, but rather about deductive rules and queries.

Negation as failure is syntactially in prolog the "\+" operator, but semantically it can behave as "~exists x1, ..., exists xn".

So a first step to arrive at a negation as failure simulation of your formula is to replace all quantifiers "forall x" by "~exists x~". Further you can move down negation and also replace implication or biconditional.

When you have done all this you can sort out "~exists x1, ..., \$ exists xn" and see to it that the variables x1..xn are positively bound. If this is the case then you are probably done.

I will try your example:

```
A x (student(x) => E y ( table(y) & on( x, y)))  
<=> ~E x ~(~student(x) v E y(table(y) & on(x,y)))  
<=> ~E x (student(x) & ~E y(table(y) & on(x,y)))  
positivity check is ok, so this amounts to prolog as:  
<=> \+ (student(x), \+ (table(y), on(x,y)))
```

The remaining issues are how student, table, on are defined. But you can not solve all issues with this approach. FOL is still a little bit more expressive in directly representing, for example when it comes to disjunctive or existential knowledge.

See also for example:

John W. Lloyd: Foundations of Logic Programming,
1st Edition Springer 1984

Thank you that is awesome, I have read a few articles on converting FOL to clauses form but don't know exactly the use of it, but after

Re: predicate in a predicate

the example you showed me, I got it now. Hehe. I will lookup for the book but I don't where to get it other than Amazon, but I think I found something like it

www.cs.utexas.edu/users/vl/mypapers/flp.ps

.. Thanks anyway.

.