Thread: How can fill the "logical variable"?

Own Daghagheleh 7 days ago

How can fill the "logical variable"?

Hi, I used hint3 and assigned a variable to each box ('_'). now I don't know how to work with logical variables? for example how to collect them as a slot and fill them? logically sounds they give lots of flexibility to the code but I don't how to use them. Appreciate your help in advance. Thanks

Reply Quote Email Author

Peter Schachte 7 days ago

RE: How can fill the "logical variable"?

Yes, good question. Unbound variables are a bit difficult to work with sometimes, because it's easy to accidentally unify them with something you don't mean to.

The easiest solution in this case is to wrap each variable in a functor, so you can distinguish them from other things. For example, instead of transforming a row like

['#','_','_','#']

to

['#',A,B,'#']

(where the A and B are just fresh variables), I recommend transforming it into

[solid,slot(A),slot(B),solid]

or something like that. That's just as easy to transform into, and then it's much easier to extract the slots from that, giving you the list

[A,B]

which is easily unified with a two-letter word. It's also easy to use the tranpose predicate on a list of the [solid,slot(A),slot(B),solid] list, and to extract the slots from that version, and most importantly, when you extract the slots from that, they will share the variables with variables in the slots you first extracted. That means that when you unify [A,B] with a word, it will bind the A and B that appear in the vertical slots.

Reply | Quote | Email Author

▲ Hide 10 replies

Junwei Yang 7 days ago

RE: How can fill the "logical variable"?

Hi Peter, But when I tried to replace underscore with logical variables, how can I make sure each underscore is replaced with different variables? For example, the first fillable slot can be replaced with fill(A), and the second fill(B), etc. My thought is to have a list of variables so that I can remove used variables to avoid repeating using the same variable. But I don't think it's a good practice to write like this. Thanks.

Reply | Quote | Email Author

▲ Hide 2 replies

Peter Schachte 7 days ago

RE: How can fill the "logical variable"?

Prolog will do that for you. Each underscore is a distinct Prolog variable; until you unify it with something else, it will remain different from other variables. If you want to substitute fill(A) for the first '_' and fill(B) for the second, and so on, just substitute fill(_) for each of them, and they'll all be different.

Reply | Quote | Email Author

▲ Hide 1 reply

Les Kitchen 6 days ago

RE: How can fill the "logical variable"?

If I can add an actual example, which might help illustrate what goes on:

```
foo(X) :-
        X = [_,_,_,_].
bar(X) :-
        L = [_,_],
        append(L, L, X).
```

foo/1 checks that its argument is a list of four anythings. bar/1 internally enforces that L be a list of two anythings, and enforces that its argument X be two of those concatenated together. See how they behave differently, when loaded into swipl:

```
?- ['wildcard.pl'].
% wildcard.pl compiled 0.00 sec, 3 clauses
true.

?- foo(X).
X = [_G2704, _G2707, _G2710, _G2713].

?- bar(X).
X = [_G9, _G12, _G9, _G12].

?- foo([a,b,E,F]).
true.

?- bar([a,b,E,F]).
E = a,
F = b.

?- foo([a,b|T]).
T = [_G277, _G280].

?- bar([a,b|T]).
T = [a, b].
```

foo(X) gives us a list of four different generated variables. bar(X) gives us a list of four generated variables, but the first and third variables are the same, as are the second and fourth (one-origin).

foo([a,b,E,F]) succeeds whatever the bindings of E and F. bar([a,b,E,F]) succeeds only with E and F bound to a and b, respectively.

Breaking the list up differently, foo([a,b|T]) succeeds with T bound to any list of two elements, while bar([a,b|T]) only with T bound to precisely the list [a,b].

More generally, every appearance of _ in a clause gives you a fresh variable, but you can end up with data-structures in which that fresh variable occurs in multiple places, constraining the values at those places to be the same. That's what can be used for Proj2 — although as Peter Schachte notes, you might need to wrap the variables inside a suitable term, to prevent matches that you don't want.

Hope this is useful. Re-post if you're still not sure.

— Smiles, Les.

P.S. **Rant warning:** Mostly irrelevant to DP itself, but relevant to computing in general, so I'll mention it here: I'm posting this, and running the Prolog examples, on one of my year-2000 vintage Pismo Powerbooks. Apple dropped MacOS support for the Pismos in 2005. So now I run current stable Debian GNU/Linux, with up-to-date software and some hardware upgrades to get good-enough performance for most work I do. That's pretty remarkable, I'd say, for a 16-year-old machine. It has very nice keyboard, really the best laptop keyboard I've ever used, very

comfortable to type on — a big plus for any serious programmer. And it's gratifying to keep a device in use, and out of recycling or landfill.

It's a complicated (and partly political) issue, but I think it's fair to say that generally, most of the consumer computing industry has an interest in getting people to buy more stuff. There's an expectation that every few years you'll need to buy a new laptop or new mobile phone, to keep up. If we had machines that could be more easily upgraded and repaired, with longer-term software support, then we'd all be better off, and there'd be less demand on Earth's resources and less electronic waste. That's my opinion. The free-software, open-knowledge, maker, and repair movements are all related parts of this picture. The Embedded Open Modular Architecture/EOMA68 is just one example.

Reply | Quote | Email Author

Own Daghagheleh 6 days ago

RE: How can fill the "logical variable"?

Thanks Peter

Reply | Quote | Email Author

Yusuf Hamzah 3 days ago

RE: How can fill the "logical variable"?

I have question regarding this. What approach would you use to differentiate between solid and slot(_)? For example: [solid, slot(A),slot(B), solid, solid, slot(C), slot(D)].

Reply | Quote | Email Author

▲ Hide 5 replies

Peter Schachte 3 days ago

RE: How can fill the "logical variable"?

Unification. Write a predicate like extract_slots that skips over solid at the front of the list, and handles slot(X) by collecting the X parts into a list.

Reply | Quote | Email Author

▲ Hide 4 replies

Michael Lee 2 days ago

RE: How can fill the "logical variable"?

Thanks for all the help Peter.

One thing I'm not sure about is if the puzzle has characters such as 'a' in a horizontal row ['#', '_', 'a', '_', '_'].

Any suggesstions how I can go about storing a vaild filled char as opposed using something like fill(_) for '_'?

Reply | Quote | Email Author

▲ Hide 3 replies

Peter Schachte 2 days ago

RE: How can fill the "logical variable"?

You transform '_' into fill(_), and transform, eg, 'e' to fill(e).

Reply | Quote | Email Author

▲ Hide 2 replies

Michael Lee 1 day ago

RE: How can fill the "logical variable"?

Thanks Peter, I've managed to get that done.
I'm still stuck at obtaining a list of a list of slots from

```
[solid, slot(A),slot(B), solid, solid, slot(C), slot(D)]
```

to get

```
[ [slot(A),slot(B)], [slot(C), slot(D)] ]
```

Am I missing something glaringly obvious as I difficulty getting to do that?

Edit: I've read the thread "**How to label the logic variables (Hint 3)**" as well but I couldn't even get the first slot as well.

Reply | Quote | Email Author

▲ Hide 1 reply

Peter Schachte 54 minutes ago

RE: How can fill the "logical variable"?

I'd suggest writing two predicates: one that skips over all the solids at the front of a list, returing either an empty list or a list that begins with a slot(_); and one that collects a run of slot(X)s, returning the list of the Xs and what comes after, either [] or a list beginning with solid. Then write a third predicate that calls those two predicates in turn, over and over.

Reply | Quote | Email Author

Select: All None

Message Actions | Expand All | Collapse All

← OK