Thread: Getting the slots from a list

Select: All None

Message Actions Expand All Collapse All

8 Posts in this Thread 0 Unread

David Alexander Tedjopurnomo 3 days ago

Getting the slots from a list

Hi all,

I've struggled for quite some time in trying to retrieve the slots from a row of the puzzle.
Say that I have a list of [A,B,#,#,C,#,D]
Retrieving the slots correctly would result in: [[A,B],[C],[D]]
However, I can't get this to work. I am not even sure if this is the right way to do it.
Am I on the right track here?

Thanks in advance,
David

Reply Quote Email Author

Yusuf Hamzah 3 days ago

RE: Getting the slots from a list

Hi,

My previous question about how to differentiate between solid and slot(A) in the discussion forum naturally flow to this.

I haven't proceed to it as David because I am not even sure whether getting the output (both either [[A,B],[C],[D]] or [[Slot(A),Slot(B)],[Slot(C)],[Slot(D)]) will help in building the project further.

Regards,

Yusuf

Reply | Quote | Email Author

▲ Hide 6 replies

Peter Schachte 3 days ago

RE: Getting the slots from a list

You should also read the "How can fill the "logical variable"?" thread for discussion of this topic.

First, a correction: [A,B,#,#,C,#,D] would only have 1 slot: [A,B]. Slots have to be at least two characters long; you should ignore 1-character runs. There just aren't enough 1-letter words for a 1 character slot to be much use.

Second, I would suggest you avoid putting variables alone in a list mixed with '#' characters. It's too messy to tell them apart, since when you unify a variable with '#' it will succeed, binding the variable to '#'. I would suggest instead using a term containing a varialbe, such as slot(A) for the slots. These will clearly not unify with '#', but they still contain the variable for you to put into the slot when you make a list of slots. And when you transpose this list of lists, it will contain the same variables, so when you make the list of vertical slots they will share variables with the horizontal slots. That's the key part of making this work.

To answer Yusuf's question, once you have a list of slots, where each slot is a list of variables or letters, and assuming you have a list of words, where each word is a list of letters, you're more than half way done. You just need to make for each slot a list of all the words that match that slot. You can use setof for that, and finding which words (from the list Words) that match a slot (in the variable Slot) is as simple as member(Slot, Words). Once you've done that, you pick one of the slots with the fewest matching words, and nondeterministically bind the slot to one of the words (for this, I would recommend using select/3, because it will also give you back the list of words with the selected word removed, so you don't use any word more than once). Now take the remaining slots and the remaining words, and recursively do the same thing. That's really all you have to do. Prolog's backtracking will take care of backing up and picking a different word to fill into the slot if your selection leave the puzzle unsolvable.

Reply | Quote | Email Author

▲ Hide 5 replies

David Alexander Tedjopurnomo 2 days ago

RE: Getting the slots from a list

Thanks for the reply, Peter.
There's one thing that I am still ambiguous about. If I find that a slot, say slot(A), matches an alphabet 'F', does that mean that I have to bind 'F' to A, ending with slot('F')?

Reply | Quote | Email Author

▲ Hide 4 replies

Peter Schachte 1 day ago

RE: Getting the slots from a list

Yes, except you're really binding A to 'F' rather than the other way around., since you can only bind a variable.

Reply | Quote | Email Author ▲ Hide 3 replies

Michael Marshall 22 hours ago

RE: Getting the slots from a list

Hey Peter, I'm having trouble figuring out how to wrap the slot functor around a variable.

Do we need to define a function 'slot(X) :- ..blah...' somehow to make this work?

Reply | Quote | Email Author ▲ Hide 2 replies

Les Kitchen 3 hours ago

RE: Getting the slots from a list

Hi Michael,

Even though Prolog is “declarative” language, you normally don't have to declare/define things like you do in Haskell.

So, in Prolog, a functor just gets implicitly defined by being used in the program. You can just use slot(...) in your program, and that's all you need to do. You just need to have a clear idea of what your functor's for, and use it consistently.

Atomic terms, like what you might call solid, are analogous. And, indeed they can be thought of as just zero-arity functors.

You can see this happening in some of the workshop examples, where you can have functors like empty/0 and node/3 for defining tree structures. They don't have to be declared explicitly; they're just declared implicitly by use.

Functors in Prolog give you a way of building arbitrary datastructures as terms (though solid/0 and slot/1 are pretty simple as datastructures go). Prolog functors are pretty much analogous to data constructors in Haskell, except that you don't need to declare them explicitly, and there's no static type checking. Maybe, though, it's helpful when you're using functors in Prolog to have in the back of your mind what the corresponding Haskell data definition would look like.

— Smiles, Les.

Reply | Quote | Email Author

Les Kitchen 3 hours ago

RE: Getting the slots from a list

And a just a followup: It wouldn't make sense to write something like slot(X) :- ..blah... in this context, since :- works with a predicate as head, but slot/1 is intended as a functor.

Prolog semantics has two very distinct levels: at one level predicates which are assertions or queries about truth (and which can be combined into clauses); and at the other level terms, which are in effect free-form datastructures built with functors. In a Prolog program, terms can appear only as arguments to predicates; they can't be free standing.

Reply | Quote | Email Author

Select: All None

Message Actions | Expand All | Collapse All

← OK