Universidade de Brasília

Departamento de Ciência da Computação

Lógica Computacional 1

Relatório de Projeto

Prova da equivalência entre diferentes definições de permutação no assistente de provas Coq

1] Andrey Calaca Resende
180062433

2]Gustavo Lopes Dezan

3]Felipe Dantas Borges
202021749

4]Eduardo Ferreira Marques Cavalcante
202006368

Professor:

Flavio Leonardo Cavalcanti de Moura

Data de envio : 03/09/2022

1. Introdução

As permutações estão presentes em diversos contextos no âmbito acadêmico, mas qual a sua definição? Como foi demonstrado ao longo do projeto, é possível criar mais de uma definição para esse termo e provar sua equivalência utilizando a lógica de primeira ordem e o auxílio do assistente de provas Coq.

A definição mais comum e aceita é de que a permutação é a troca da ordem de elementos dentro de um contexto. Nesse caso, podemos ser mais específicos e dizer que uma lista de números naturais é permutação de outra se essas possuírem os exatos mesmos elementos independente da ordem em que eles aparecem. Dessa forma, se cada elemento da lista A possui uma correspondência exata na lista B, então as listas A e B são permutações uma da outra.

Utilizando a notação matemática aceita pelo Coq podemos escrever essa afirmação da seguinte forma:

```
Definition equiv l l' := forall n:nat, num_oc n l = num_oc n l'.
```

Esse código indica que dadas duas listas de números naturais l e l' elas serão equivalentes, ou permutações uma da outra, se cada número natural possuir o mesmo número de ocorrências em l e em l'. Aqui utilizamos a função num_oc para contar essas ocorrências em cada lista.

Porém, também é possível criar uma definição para permutação utilizando o princípio da indução da seguinte maneira:

```
Inductive perm : list nat -> list nat -> Prop :=
| perm_eq: forall l1, perm l1 l1
| perm_swap: forall x y l1, perm (x :: y :: l1) (y :: x :: l1)
| perm_hd: forall x l1 l2, perm l1 l2 -> perm (x :: l1) (x :: l2)
| perm_trans: forall l1 l2 l3, perm l1 l2 -> perm l2 l3 -> perm l1 l3
```

O restante do projeto visa demonstrar como podemos provar a equivalência entre as definições de permutação utilizando táticas fornecidas pelo Coq para simplificar a linguagem matemática da Lógica de Primeira Ordem.

2. Procedimentos

Para provar a equivalencia entre diferentes definicoes de permutacao foi utilizado processo de dedução natural auxiliado pela ferramenta de software COQ[1].Por meio do software, foi escrito um arquivo de provas "PermEquiv.v" o qual contém as definições necessárias e todo o sequenciamento lógico desenvolvido.

A equivalência foi feita da seguinte forma:

1. Primeiramente, mostrar que o Permutation do coq library possui equivalência com perm deste trabalho uma verificação se uma permutação "perm" indutiva de duas listas l e l' implica em uma permutação "permutation" das mesmas listas l e l'.
2. Segundo, equiv $\Leftrightarrow perm$:

A verificação se uma permutação quantitativa de duas listas l e l' apresenta bi-implicação em uma permutação indutiva das mesmas listas l e l'. Através da prova de lemas principais e auxiliares para essas implicações.

Utilizamos de alguns lemmas auxiliares que nos ajudaram a realizar as provas de forma mais concisa e separadamente. como se pode ver a seguir:

1. `Lemma equiv_nil: ∀ l, equiv nil l → l = nil.` Possível de se provar com a regra $(permutation_nil)$.
2. `Lemma num_oc_neq: ∀ n a l₁ l₂, n =? a = false → num_oc n (l₁ ++ a :: l₂) = num_oc n (l₁ ++ l₂).` é usado para concluir que se tem tem uma hipótese em que a conclusão é igual ao equiv to perm
3. `Lemma num_occ_cons: ∀ l x n, num_oc x l = S n → ∃ l₁ l₂, l = l₁ ++ x :: l₂ ∧ num_oc x (l₁ ++ l₂) = n.` o lema tem propriedades em que uma lista l pode ser reescrita, sabendo que x ocorre ao menos uma vez em l.
4. `Lemma num_oc_S: ∀ x l₁ l₂, num_oc x (l₁ ++ x :: l₂) = S (num_oc x (l₁ ++ l₂)).` é usado no equiv to perm sabendo que os dois termos são equivalentes e quer transformar um no outro.
5. `Lemma perm_app_cons: ∀ l₁ l₂ a, perm (a :: l₁ ++ l₂) (l₁ ++ a :: l₂).` Lema sobre a reorganição de uma lista l.
   baseado nas notas de aula da disciplina de Lógica Computacional 1, [2] e pela prova disponível em [3].

3. Formalização

Para efetuar a formalizaçao deste processo foi utilizado a lógica de primeira ordem, para que dessa forma houvesse aplicação do processo indutivo nas estruturas lógicas, o qual foi realizado com o uso do já citado Coq, sendo um sistema formal de gerenciamento de provas. Ele fornece uma linguagem formal para escrever definições matemáticas, algoritmos executáveis e teoremas juntamente com um ambiente para desenvolvimento semi-interativo de provas verificadas por máquina.

4. Conclusao

Por fim, com as implicação provados, se torna possível, dessa forma, concluir a equivalência entre as duas noções de permutação desejadas, mostrando que é possível provar a equivalência entre uma definição formal indutiva e uma não indutiva . Em seguida, foi possível concluir que é facilitado o processo de prova por indução quando a análise decorre em estruturas indutivamente definidas e se complica caso contrário. Além disso, percebe-se que o processo de provas computacionalmente no Coq apresenta alta complexidade de compreensão, demandando maior atenção, do qual foi utilizado o site Stackoverflow[4] para tirar dúvidas. Entretanto, como retorno, foi possível adquirir um maior conhecimento quanto ao software de auxílio de provas, formalização e dedução natural.

5. Referências

References

[1] “List permutations as a composition of adjacent transpositions,” Disponível em: https://coq.inria.fr/library/Coq.Sorting.Permutation.html, [Online].

[2] F. Moura, “Lógica computacional e algoritmos,” Disponível em: http://flaviomoura.info/files/lca.pdf, 2022, August, [Online].

[3] ——, “Permequiv,” Disponível em: https://github.com/ensino-unb/PermEquiv/blob/main/PermEquiv.v, 2021, August, [Online].

[4] “I’m trying to build a proof in coq that two different permutation definitions are equivalent, but the non-inductive side is not working,” Disponível em: https://stackoverflow.com/questions/73557184/im-trying-to-build-a-proof-in-coq-that-two-different-permutation-definitions-ar, 2022, September, [Online].

[5] H. Anoun, “Equivalence between two definitions of permutations of lists,” Disponível em: http://www.cse.chalmers.se/research/group/logic/TypesSS05/resources/coq/CoqArt/newstuff/permutations.html, [Online].

[1] [4] [3] [2] [5]