Relatório Final - Lógica Computacional 1[1]

Marcelo Araújo - (150016794), Augusto Brandão - (160024366), Ian Nery - (150129670)

[1]Departamento de Ciência da Computação – Universidade de Brasília (UNB)

Introdução

A lógica computacional, extenso campo de pesquisa e aplicações práticas, é primordialmente utilizada para a comprovação do funcionamento de sistemas críticos hodiernos, que a torna essencial tanto como especialização na área, quanto como introdução para outros âmbitos de pesquisa, como a de inteligência artificial. O objetivo do projeto em questão é principalmente introduzir os alunos a ferramentas de verificação e formalização para a melhor absorção e entendimento das técnicas de manipulação lógica apresentadas em sala, além de demonstrar como tais conceitos aprendidos são aplicados na prática.

No projeto contido nesse relatório, é apresentado para a prova questões sobre propriedades envolvendo a inserção e ordenação de sequências. A ferramenta utilizada para a especificação e verificação formal da lógica por detrás de tais conjecturas é o PVS. O software mencionado é de grande importância para as aplicações práticas descritas acima e outras, como é possível constatar pelo diretório da NASA, que abarca diversas bibliotecas acerca de diversas conjecturas similares a apresentada no trabalho, com a finalidade de auxiliar os pesquisadores da área a ter um entendimento similar ao proposto pelo projeto.

Explicação das soluções

Questão 1

A conjectura da questão 1, *fs_insert_in_sorted_preserves_sort* propõe que para todo s sequência finita e x natural, têm-se que a ordenação de s implica na ordenação de s após uma inserção de x em si. Para o início de sua prova, fora ultilizado um passo de indução forte para a sequência "s" e para a variável "length(s)", sendo essa a variável que define o tamanho da sequência finita "s". A estratégia primordial para a prova de tal conjectura foi a utilização da skolemização e instanciação, relatadas a regras referentes aos quantificadores universais da lógica de Gentzen; manipuladores equacionais com a finalidade de dissecar o problema para torná-lo trivial, como enunciado pelos comandos **lift-if**, **replace** e **expand**; manipuladores lógico-proposicionais, como **prop**; e também comandos com a função de cut, como **case** e **lemma**.

Questão 2

A questão 2, *fs_insertionsort_is_sorted*, é uma conjectura cuja proposição é: para toda sequência finita natural "s" têm-se que a inserção ordenada em "s" torna-se em uma sequência ordenada. Para a prova da conjectura, foi utilizado como na questão 1 a estratégia de indução forte, para as mesmas "s" e "length(s)", e além dos comandos de skolemização e instanciação, foram utilizados novamente os manipuladores equacionais **lift-if**, **replace** e **expand**; o manipulador lógico-proposicional **prop**; e por fim os comandos que exprimem a função cut, nesse caso **case**, **rewrite** e **lemma**.

Questão 3

A questão 3, *fs_ins_and_add_in_perm_is_perm*, consiste na proposição que para quaisquer as sequências ”s1” e ”s2”, que são permutações entre si, e para qualquer variável ”x”, têm que que a inserção de x no primeiro elemento da sequência s1 e a inserção de x dada pelo método *insertion* em s2, s1 e s2 ainda são permutações. O método para prova da conjectura foi o uso dos mesmos mecanismos da questão 2, além do comando **typepred**, cuja função é de apresentar restrições de tipo em expressões previamente estabelecidas.

Questão 4

Já a questão 4, *fs_insertionsort_is_permutations*, representa que para cada sequência ”s”, a inserção ordenada em s e a sequência original s sofrem permutação. Na prova feita, foi utilizado o passo de indução forte como na questão 1, os métodos descritos nas questões 2 e 3, e também o comando de manipulação equacional **decompose-equality**, cuja função é decompor igualdades em componentes mais triviais para prova, como a decomposição de uma igualdade de sequência a separa em uma igualdade de sequências junto a uma igualdade de seus tamanhos.

Especificação do problema e explicação do método de solução

Nesta etapa mostraremos o desenvolvimento das provas e suas relações com a dedução natural no cálculo de Gentzen. Omitiremos certos passos com a finalidade de manter a prova legível, como certos comandos de $expand$, passos omitidos serão sinalizados adequadamente quando necessários, além disso, fórmulas escondidas com a mesma finalidade estarão representadas como Γ do lado esquedo do sequente e como Δ do lado direito do sequente.

Questão 1

Usaremos algumas abreviaçõe com o intuito de diminuir os tamanhos das provas assim, $s?$ abrevia $is.sorted?(x\ :\ finseq)$, i abrevia $insertion(x\ :\ nat, y\ :\ finseq)$, l abrevia $length(x\ :\ finseq)$, s abrevia $seq(.)$, af abrevia $addfirst(x\ :\ nat, y\ :\ finseq)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\nabla_1 \qquad \nabla_2 \qquad \nabla_3}{\Gamma, s?(x_1) \Rightarrow \Delta_1}\,(prop)}{\Gamma, s?(x_1) \Rightarrow s?(i(x, x_1))}\,(R_=)}{\Gamma \Rightarrow s?(x_1) \rightarrow s?(i(x, x_1))}\,(R_\rightarrow)}{\dfrac{\forall_y \forall_x\ (l(y) < l(x_1)) \rightarrow s?(y) \rightarrow s?(i(x, y)) \Rightarrow \forall_x s?(x_1) \rightarrow s?(i(x, x_1))}{\Rightarrow \forall_{s,x}\ s?(s) \rightarrow s?(i(x, s))}\,(mi+)}\,(R_\forall)\ e\ (LW)$$

Onde Δ_1 foi usado para abreviar a igualdade usada por $(R_=)$, aplicada pelo comando $expand$, as regras de $(R_\forall)$ e $(R_\rightarrow)$ foram aplicadas com o comando $skeep$.

$$\begin{aligned} \Delta_1 = IF\ l(x_1) = 0\ THEN s?(af(x, x_1)) \\ ELSE\ IF\ x \leq f(x_1)\ THEN\ s?(af(x, x_1)) \\ ELSE\ s?(af(f(x_1), i(x, rest(x_1)))) \end{aligned} \tag{1}$$

Agora podemos explicar o uso do comando *prop* em nossa prova para obtermos os ramos $\nabla_{1,2,3}$, a estrutura de $if, then\ else$ pode ser interpretada como uma série de implicações, como no exemplo de Δ_1, temos, $l(x_1) = 0 \to s?(af(x, x_1))$, como uma primeira implicação, $(\neg(l(x_1) = 0) \wedge x \leq f(x_1)) \to s?(af(x, x_1))$, e temos uma terceira implicação que correspode ao caso que nenhuma das duas condições dos $if's$ são verdadeiras, $(\neg((l(x_1) = 0) \wedge x \leq f(x_1))) \to s?(af(x, x_1))$.

Utilizando o comando *prop* nessa estrutura obtemos 3 ramos, provenientes da utilização de comandos como $flatten$, *split* sucetivamente, assim expandindo o *prop* utilizado acima podemos ver as seguintes provas:

$$\dfrac{\dfrac{\nabla_1'}{\Gamma, l(x_1) = 0, s?(x_1) \Rightarrow s?(af(x, x_1))} \qquad \nabla_2 \qquad \nabla_3}{\Gamma, s?(x_1) \Rightarrow \Delta_1}\ (prop)$$

$$\dfrac{\nabla_1 \qquad \dfrac{\nabla_2'}{\Gamma, x \leq f(x_1), s?(x_1) \Rightarrow s?(af(x, x_1)),\ l(x_1) = 0} \qquad \nabla_3}{\Gamma, s?(x_1) \Rightarrow \Delta_1}\ (prop)$$

$$\dfrac{\nabla_1 \qquad \nabla_2 \qquad \dfrac{\nabla_3'}{\Gamma, s?(x_1) \Rightarrow s?(af(x, x_1)),\ l(x_1) = 0, x \leq f(x_1)}}{\Gamma, s?(x_1) \Rightarrow \Delta_1}\ (prop)$$

Os comando *split* aplica as regras de $L_\vee$, $R_\wedge$, e $R_\to$, e o comando $flatten$ as regras de $R_\vee$, $L_\wedge$, e $L_\to$, e vemos que são essas exatas regras aplicadas aos sequentes da questão em si.

Assim, isso nos dá 3 objetivos para provar, o primeiro representado por ∇_1', onde queremos provar que para uma sequência x_1 de tamanho 0, vazia, a adição de um natural x em uma lista vazia resultam em uma lista ordenada, no caso a lista resultante tera apenas um elemento, estando ordenada, a prova deste ramo pode ser concluída sem maiores dificuldades com os comando apresentados.

Para o ∇_2' temos o objetivo de provar que para uma lista x_1 que está ordenada, com tamanho diferente de 0, a adição de um elemento menor ou igual ao primeiro elemento de x_1 no indíce 0 resulta em uma lista ordenada. Podemos ver que se a lista está ordenada e o elemento a ser inserido e menor ou igual ao primeiro de x_1 adicioná-lo na primeira posição mantém a ordenação.

$$\dfrac{\dfrac{\dfrac{\nabla_4}{\Gamma \Rightarrow k = 0, \Delta} \qquad \dfrac{\nabla_5}{\Gamma, k = 0 \Rightarrow \Delta}}{\Gamma, \forall_k k \leq l(x_1) - 2 \to x_1.s(k) \leq x_1.s(k+1) \Rightarrow af(x, x_1).s(k) \leq af(x, x_1).s(k+1), \Delta}\ (Cut)}{\dfrac{\vdots}{\Gamma, x \leq f(x_1), s?(x_1) \Rightarrow s?(af(x, x_1)),\ l(x_1) = 0}}$$

Interessante desta prova é a utilização da regra de Cut, no caso utilizamos $k = 0$, onde k seria o índice de acesso na sequência, assim teremos um ramo provado por ∇_5 que indica que o valor de k é 0 oque nos dá realizando expansões que o primeiro elemento,

$af(x, x_1).s(0) \leq af(x, x_1).s(1)$, expandindo temos $x \leq x_1.s(0)$ do lado direito do sequente, e temos exatamente isso do lado esquerdo também, dentro do Γ, assim fechamos este ramo.

$$\Gamma, x \leq x_1.s(0) \Rightarrow x \leq x_1.s(0), \Delta \quad (Ax) \tag{2}$$

Para ∇_4 temos um quantificador universal a direita, $\forall_k k \leq l(x_1) - 2 \rightarrow x_1.s(k) \leq x_1.s(k+1)$, e agora precisamos usá-lo para provar que os elementos de índice k são menores que $k+1$ com k maior que 0, podemos notar que com esse íncide maior que 1 para a lista nova, $af(x, x_1)$ temos os mesmos elementos da lista x_1, que está ordenada, $\Gamma, s?(x_1)$. Utilizamos o comando $inst(\text{“}k - 1'')$ para remover o quantificador gerado pela definição de is_sorted?, $(L_\forall)$. Também temos que remover a implicação presente na fórmula do antecedente, usamos a regra $(L_\rightarrow)$, assim temos mais 2 objetivos para provar, um objetivo provando que o valor de k é valido, e definitivamente que a lista em si está ordenana, como mencionado anteriormente.

$$\dfrac{\nabla_i \qquad \dfrac{\nabla_j}{\Gamma, x_1.s(k-1) \leq x_1.s(k) \Rightarrow k = 0,\ af(x, x_1).s(k) \leq af(x, x_1).s(k+1), \Delta}}{\Gamma, (k-1) \leq l(x_1) - 2 \rightarrow x_1.s(k-1) \leq x_1.s(k) \Rightarrow \Delta} L_\rightarrow$$

$$\dfrac{\dfrac{\nabla_i'}{\Gamma, (k-1) \leq l(x_1) - 2 \Rightarrow k - 1 \leq l(x_1) - 2} \qquad \nabla_j'}{\Gamma, (k-1) \leq l(x_1) - 2 \rightarrow x_1.s(k-1) \leq x_1.s(k) \Rightarrow \Delta} L_\rightarrow$$

Agora para terminarmos a prova da questão 1 precisamos apenas provar ∇_3', utilizando a nossa hípotese de indução gerada pelo $measure - induct+$ que estava omitida em Γ, $\forall_y \forall_x\ (l(y) < l(x_1)) \rightarrow s?(y) \rightarrow s?(i(x, y))$, utilizamos a regra $(L_\forall)$ duas vezes entao teremos:

$$\dfrac{\dfrac{\dfrac{\nabla_6}{\Gamma \Rightarrow (l(r(x_1)) < l(x_1)), \Delta} \qquad \dfrac{\nabla_7}{\Gamma, s?(x_1) \Rightarrow s?(r(x_1))\Delta} \qquad \dfrac{\nabla_8}{\Gamma, s?(i(x, r(x_1))) \Rightarrow \Delta}}{(l(r(x_1)) < l(x_1)) \rightarrow s?(r(x_1)) \rightarrow s?(i(x, r(x_1))) \Rightarrow \Delta} (L_\rightarrow)}{\Gamma, \forall_y \forall_x\ (l(y) < l(x_1)) \rightarrow s?(y) \rightarrow s?(i(x, y)) \Rightarrow s?(af(f(x, x_1), i(x, r(x_1)))), \Delta} (L_\forall)^{2\times}$$

∇_6 consiste em provar que o tamanho do resto de x_1 é menor que o tamanho de x_1, pode ser provado com algumas expansões e o lemaa $empty_seq$, que é a aplicação da regra (Cut).

∇_7 consiste em provar que se x_1 está ordenada o resto de x_1 também está.

E então o caso mais complicado, ∇_8 se resume em provar que se o natural x foi inserido em $r(x_1)$ e permaneceu ordenado, a adição do $f(x_1)$ nesta lista inserida mantém a propriedade de ordenada. Utilizamos a regra Cut, usando $l(r(x_1) = 0)$, assim temos 2 objetivos, provar que se isto for verdade temos uma lista de 2 elementos $[f(x_1), x]$ está ordenada, como sabemos que $x > f(x_1)$ isso está correto, para o segundo objetivo vai se resumir, com algumas manipulações em provar que $f(x_1) \leq f(i(x, r(x_1)))$, assim temos que provar para os casos de x ter sido inserido no primeiro índice do resto de x_1, assim teremos $f(x_1) \leq x$, e para o caso em que x foi inserido após isso tendo, $f(x_1 \leq f(r(x_1)))$, com as inforções que temos de $s?(x_1)$ e $s?(i(x, r(x_1)))$ a prova pode ser concluída fácilmente.

$$\dfrac{\dfrac{\dfrac{\nabla}{\Gamma \Rightarrow f(x_1) \leq f(i(x, r(x_1))), \Delta} \qquad \dfrac{\nabla}{\Gamma \Rightarrow f(x_1 \leq f(r(x_1))), \Delta}}{\vdots \atop \Gamma \Rightarrow l(r(x_1)) = 0, \Delta} \qquad \dfrac{\nabla}{\Gamma, l(r(x_1)) = 0 \Rightarrow \Delta}}{\Gamma, s?(i(x, r(x_1))), s?(x_1) \Rightarrow s?(af(f(x, x_1), i(x, r(x_1)))), \Delta}\ Cut$$

Questão 2

Utilizaremos novamente as abreviações: $s?$ abrevia $is.sorted?(x : finseq)$, i abrevia $insertion(x : nat, y : finseq)$, l abrevia $length(x : finseq)$, s abrevia $seq(.)$, f abrevia $first(y : finseq)$, fis abrevia $(fs\_insertion\_sort(x : finseq))$, r abrevia $(rest(x : finseq))$, também sabendo que min, p são funções decorrentes da manipulação equacional, e $eseq$ o lema $empty_seq$

$$\dfrac{\dfrac{\dfrac{\nabla_1 \qquad \nabla_2}{\forall_y\ l(y) < l(x!1) \rightarrow s?(fis(y)) \Rightarrow \Delta_1}\ (prop)}{\forall_y\ l(y) < l(x!1) \rightarrow s?(fis(y)) \Rightarrow s?(fis(x!1))}\ (R_=)}{\forall_y\ s?(fis(s))}\ (mi+)$$

Sendo Δ_1 o resultado da manipulação equacional da expressão $s?(fis(x!1))$, demonstrada na árvore pelo símbolo $(R_=)$, tal que descreve a seguinte função:

$$\begin{aligned} \Delta_1 = IF\ l(x_1) = 0\ THEN s?(x!1)) \\ ELSE\ s?(i(f(x!1))), fis(r(x!1)) \end{aligned} \tag{3}$$

Após a utilização do comando $prop$ na prova, temos as proposições definidas pelos ramos $\nabla_{1,2}$, e como podemos interpretar a estrutura $if, thenelse$ como séries de implicações, temos que ∇_1 exprime a proposição $l(x!1) = 0 \ \rightarrow \ s?(x!1)$, e na proposição de ∇_2 temos que $\neg(l(x!1) = 0) \ \rightarrow \ s?(i(f(x!1))), fis(r(x!1))$.

Temos que a continuação do galho de ∇_1:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\bot \Rightarrow}{k \ \leq \ -2 \ \Rightarrow}\ L_\bot}{l(x!1) = 0,\ k \leq 0 - 2 \ \Rightarrow \ x!1‘seq(k) \ \leq \ x!1‘seq(k+1)}\ RW, LW}{l(x!1) = 0,\ k \leq l(x!1) - 2 \ \Rightarrow \ x!1‘seq(k) \ \leq \ x!1‘seq(k+1)}\ L_=}{l(x!1) = 0,\ k \leq l(x!1) - 2 \ \Rightarrow \ finseq\_appl(x!1)(k) \ \leq \ finseq\_appl(x!1)(k+1)}\ R_=}{l(x!1) = 0 \ \Rightarrow \forall_{k < l(x!1)}\ k \ \leq \ (l(x!1) - 2) \ \rightarrow \ finseq\_appl(x!1)(k) \ \leq \ finseq\_appl(x!1)(k+1)}\ R_\forall}{l(x!1) = 0 \ \Rightarrow s?(x!1)}\ R_=$$

Tendo como observação que o uso da regra sequente $R_=$ como manifestação dos comandos de manipulação equacional **expand, replace** e o uso da regra sequente $R_\forall$ como representação do comando **skeep**.

Já para o galho de ∇_2, temos:

$$\dfrac{\dfrac{\dfrac{\forall_{y:finseq,\ x:nat}\ s?(y)\to s?(i(x,s))}{s?(fis(r(x!1)))\ \Rightarrow\ l(x!1)=0,\ s?(i(f(x!1)),fis(r(x!1)))}\,cut \qquad \nabla_3}{l(r(x!1))<l(x!1)\to s?(fis(r(x!1)))\ \Rightarrow\ l(x!1)=0,\ s?(i(f(x!1)),fis(r(x!1)))}\,prop}{\forall_y\ l(y)<l(x!1)\to s?(fis(y))\ \Rightarrow\ l(x!1)=0,\ s?(i(f(x!1)),fis(r(x!1)))}\,L_\forall$$

Enquanto a ∇_3 se expande para:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{1>x!1\text{‘}l-1\Rightarrow\perp}{\Delta_5}\,R_\perp \qquad \dfrac{1\geq x!1\text{‘}l-1\Rightarrow\perp}{\Delta_6}\,R_\perp \qquad \dfrac{\dfrac{\Rightarrow\perp}{\Rightarrow x!1\text{‘}l-1<l(x!1)}\,R_\perp}{\Delta_7}\,R_=,RW}{\Rightarrow x!1\text{‘}l=0,\Delta_4}\,prop}{\Rightarrow x!1\text{‘}l=0,l(x!1)=0,\ \Delta_3}\,R_=,RW}{\Rightarrow \Delta_2,\ l(x!1)=0}\,prop}{\Rightarrow\ l(r(x!1))<l(x!1),\ l(x!1)=0}\,R_=}{\Rightarrow\ l(r(x!1))<l(x!1),\ l(x!1)=0,\ s?(i(f(x!1)),fis(r(x!1)))}\,RW$$

Sendo

$$\begin{aligned}\Delta_2 = IF\ x!1\text{‘}l=0\ THEN\ x!1\text{‘}l\\ ELSE\ (x!1\ p\ (1,x!1\text{‘}l-1))\text{‘}l\end{aligned} \tag{4}$$

,

$$\Delta_3 = (x!1\ p\ (1,x!1\text{‘}l-1))\text{‘}l<l(x!1) \tag{5}$$

,

$$\begin{aligned}\Delta_4 = IF\ 1>x!1\text{‘}l-1\ OR\ 1\geq x!1\text{‘}l\ THEN\ eseq\text{‘}l\\ ELSE\ m(x!1\text{‘}l-1,x!1\text{‘}l-1)<l(x!1)\end{aligned} \tag{6}$$

$$\Delta_5 = 1>x!1\text{‘}l-1\Rightarrow eseq\text{‘}l<l(x!1),\ x!1\text{‘}l=0 \tag{7}$$

$$\Delta_6 = 1\geq x!1\text{‘}l-1\Rightarrow eseq\text{‘}l<l(x!1),\ x!1\text{‘}l=0 \tag{8}$$

$$\Delta_7 = \Rightarrow 1>x!1\text{‘}l-1\ ,\ 1\geq x!1\text{‘}l\ ,\ x!1\text{‘}l=0\ ,\ m(x!1\text{‘}l-1,x!1\text{‘}l-1)<l(x!1) \tag{9}$$

Conclusões

A partir da prova lógica das quatro conjecturas auxiliares apresentadas, é possível provar a funcionalidade da função *fs_insertion_sort* empregada como lema no apêndice do trabalho, visto que as questões posteriormente demonstradas são propriedades referentes ao mecanismo de inserção e identificação da ordenação das sequências apresentadas, a prova aplicada da função citada se compõe dos comandos de skolemização da expressão seguida da instanciação do quantificador existencial em variáveis já presentes na expressão sucedida, e então o comando de cut **rewrite** para aplicação dos lemas *fs_insertionsort_is_sorted* e *fs_insertion_sort_is_permutations*; sendo tais lemas as conjecturas provadas posteriormente pelas questões 2 e 4.

Apesar dos problemas do trabalho designado serem de relativa simplicidade para alguém especializado na área, estes foram fator determinante para a percepção dos integrantes sobre a importância e abrangência de usos da dedução natural e cálculo sequente, independente do software utilizado para a aplicação da teoria.

Referências

- Mauricio Ayala-Rincón and Flávio L.C de Moura, Applied Logic for Computer Scientists, computacional deduction and formal proffs, 2017 , chapters 3-4, pages 107-192
- PVS, Prover Guide, pvs.csl.sri.com/doc/pvs-prover-guide.pdf