Lógica Computacional I

Flávio Moura

Henrique P. S. Leite 17/0144534

Felipe Pinheiro

Gabriel Teixeira

Questão 1

Solicita-se a demonstração da conjectura *d_digits_gt*, definida em PVS na forma:

d_digits_gt : CONJECTURE

FORALL (n : nat):

10 ^ (n_digits(n)) > n

ou, em notação de Gentzen:

|-------

{1} FORALL (n: nat): 10 ^ (n_digits(n)) > n

Em linguagem natural, isso poderia ser traduzido como "para qualquer número n pertencente ao domínio dos números naturais, é verdadeira a proposição 'o resultado da exponenciação de 10 pelo número n é estritamente maior do que esse mesmo número n' ".

Damos início à prova aplicando indução forte em 'n' - isto é, provando que no passo indutivo que se a proposição é válida para um y qualquer menor do que 'n' ela será válida para 'n' - o que aparece no PVS (em notação que representa o cálculo de Gentzen) na forma de sequente abaixo:

{-1} FORALL (y: nat): y < x!1 IMPLIES 10 ^ (n_digits(y)) > y

 |-------

{1}   10 ^ (n_digits(x!1)) > x!1

Em seguida, expandimos a definição de n_digits(y) - que já estava pré-definida com um "lema" (em geral, um resultado parcial ou regra derivada que pode ser usada em outras provas – nessa questão, corresponde à definição de uma função) no arquivo fornecido – por meio do comando lemma, chegando a um comando IF que separa a hipótese para n < 10 e n > 10. Escondendo a própria definição do lema (por meio de um delete -1), chegamos a:

[-1] FORALL (y: nat): y < x!1 IMPLIES 10 ^ (n_digits(y)) > y

|-------

[1]   10 ^ (n_digits(x!1)) > x!1

Expandindo agora a definição da função do cálculo do número de dígitos de 'n' na conclusão por meio de um expand n_digits, chega-se a:

d_digits_gt :

[-1] FORALL (y: nat): y < x!1 IMPLIES 10 ^ (n_digits(y)) > y

 |-------

{1}   10 ^ IF x!1 < 10 THEN 1 ELSE 1 + n_digits(ndiv(x!1, 10)) ENDIF > x!1

Em seguida, levamos o IF à "raiz" da conclusão para que possamos separá-lo em dois sequentes distintos, que nos conduzirão a dois casos e, portanto, ramos da prova, por meio do comando *lift-if* 1 seguido do comando *prop*, chegando a:

{-1} x!1 < 10

[-2] FORALL (y: nat): y < x!1 IMPLIES 10 ^ (n_digits(y)) > y

|-------

{1}   10 ^ 1 > x!1

"Escondendo-se" a segunda premissa por meio do comando hide - 2, como se fôra um "contexto" na aplicação de uma regra **Lw** do cálculo de Gentzen, isolamos as fórmulas ativas que nos interessam na demonstração do primeiro "ramo" da prova:

d_digits_gt.1 :

[-1] x!1 < 10

|-------

[1]   10 ^ 1 > x!1

O que já é, a propósito, patentemente verdadeiro - um "axioma" no cálculo de Gentzen - embora ainda requeira a aplicação de comandos adicionais no assistente de prova. Assim, depois de um expand "^" e de dois expand "expt", chega-se a:

d_digits_gt.1 :

[-1] x!1 < 10

|-------

{1}   10 > x!1

O que pode, finalmente, ser reconhecido pelo PVS como um axioma por meio do comando "*assert*", completando o primeiro ramo, ou caso, da prova.

Para completar o segundo ramo (*subgoal*), também expandiremos as suas definições com expand “^” e expand “expt” e traremos o IF à “raiz” do sequente com o uso de *lift-if*, chegando a:

```
[-1]  FORALL (y: nat): y < x!1 IMPLIES 10 ^ (n_digits(y)) > y
|-------
[1]   x!1 < 10
{2}   IF 1 + n_digits(ndiv(x!1, 10)) >= 0
      THEN IF 1 + n_digits(ndiv(x!1, 10)) = 0 THEN 1 > x!1
           ELSE 10 * expt(10, n_digits(ndiv(x!1, 10))) > x!1
        ENDIF
ELSE 1 / (10 * expt(10, -(1 + n_digits(ndiv(x!1, 10))) - 1)) > x!1
ENDIF
```

Note-se que temos duas fórmulas na conclusão, que podem ser separadas em dois sequentes no cálculo de Gentzen por meio de um “**R^” (conjunção à direita)**, representada no PVS pelo comando *split*:

```
d_digits_gt.2.1 :
[-1]  FORALL (y: nat): y < x!1 IMPLIES 10 ^ (n_digits(y)) > y
|-------
{1}   1 + n_digits(ndiv(x!1, 10)) >= 0 IMPLIES
     IF 1 + n_digits(ndiv(x!1, 10)) = 0 THEN 1 > x!1
     ELSE 10 * expt(10, n_digits(ndiv(x!1, 10))) > x!1
     ENDIF
[2]   x!1 < 10
```

Ora, em {1} aparece uma premissa de implicação que pode ser trazida "à esquerda" para o antecedente no cálculo de Gentzen pela aplicação da regra **R->**, o que é feito por meio do comando *flatten 1* do PVS. Isso deixa um IF na conclusão que pode ser novamente quebrado em dois casos por meio da aplicação da regra **R^** de Gentzen, ou *split* em PVS, chegando-se a:

d_digits_gt.2.1.1 :

[-1] 1 + n_digits(ndiv(x!1, 10)) >= 0

[-2] FORALL (y: nat): y < x!1 IMPLIES 10 ^ (n_digits(y)) > y

|-------

{1} 1 + n_digits(ndiv(x!1, 10)) = 0 IMPLIES 1 > x!1

[2] x!1 < 10

Interessa-nos chegar a essa conclusão para "isolar" a premissa 1 + n_digits(ndiv(x!1, 10)) = 0, claramente absurda, tanto mais porque se fala aqui de números naturais (chega-se quando menos a 1 + 0 para um n<10). Essa premissa pode ser levada para o lado antecedente, de modo a chegarmos ao axioma correspondente à "eliminação do absurdo" no cálculo de Gentzen, ou "**L**⊥". Para isso, usaremos de novo o comando *flatten*, que aplica em PVS a regra **R->**, para a fórmula 1 da conclusão, o que a traz para a esquerda e nos leva a:

d_digits_gt.2.1.1 :

{-1} 1 + n_digits(ndiv(x!1, 10)) = 0 [absurdo!]

[-2] 1 + n_digits(ndiv(x!1, 10)) >= 0

[-3] FORALL (y: nat): y < x!1 IMPLIES 10 ^ (n_digits(y)) > y

|-------

{1} 1 > x!1

[2] x!1 < 10

Situação que corresponde ao axioma “**L**⊥” no cálculo de Gentzen.

tr