PCS3838 - Inteligência Artificial Exercício Prático 1

Lucas de Menezes Cavalcante - NUSP: 10770180 Engenharia da Computação, Escola Politécnica da Universidade de São Paulo

Este relatório tem como objetivo documentar a implementação em Prolog de um resolvedor de quebra-cabeças cripto-aritméticos, com a resolução específica das equações SEND + MORE = MONEY e POINT + ZERO = ENERGY.

I. Introdução

ESTE relatório tem como objetivo documentar a implementação em Prolog de um resolvedor de quebracabeças cripto-aritméticos, nos quais tem-se uma relação aritmética com letras, sendo que cada uma delas representa um algarismo diferente. Assim, deve-se relacionar valores a essas letras, de tal maneira que a equação matemática seja verdadeira.

Em particular, serão resolvidos dois problemas: o SEND + MORE = MONEY, como indicado na Figura 1, e o POINT + ZERO = ENERGY, indicado na Figura 2.

Fig. 1. Esquematização do problema SEND + MORE = MONEY.

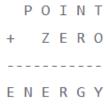


Fig. 2. Esquematização do problema POINT + ZERO = ENERGY.

Entre os quebra-cabeças cripto-aritméticos, os dois problemas apresentados possuem relevância em particular por ter uma solução única, dado que os primeiros algarismos do número (S e M, no caso do problema SEND + MORE = MONEY, e P, Z e E, no caso do problema POINT + ZERO = ENERGY) devem ser diferentes de 0.

O código para a resolução de ambos os problemas foi feito na linguagem Prolog, uma linnguagem de programação na qual utiliza-se um conjunto de restrições e relações entre variáveis para se computar algo, baseando-se na lógica de predicados [1].

II. ABORDAGEM PROPOSTA

Para a resolução do exercício em questão na linguagem Prolog, o problema foi dividido em etapas, cada qual pode ser resolvida com um conjunto de predicados na linguagem.

Inicialmente, é importante formular as restrições referentes aos valores das variáveis. Como todas são algarismos, garantese que estão todas contidas entre 0 e 9. No caso das variáveis S e M, no entanto, que são os primeiros algarismos dos números em questão, é importante que eles também sejam diferentes de 0. Assim, tem-se:

$$E, N, D, O, R, Y \in [0, 9]$$

 $S, M \in [1, 9]$ (1)

Outro requisito, como indicado previamente, é que todos os algarismos devem ser diferentes entre si.

$$S \neq E \neq N \neq D \neq M \neq O \neq R \neq Y \tag{2}$$

Por fim, deve-se satisfazer a relação matemática. Para isso, considerou-se quatro contas individuais, nas quais aparecem variáveis adicionais referentes ao carry, também conhecido como vai-um. Estas foram enumeradas em ordem, com o carry mais significativo sendo denominado C1, e o menos significativo sendo denominado C3.

$$D + E = 10 * C3 + Y$$

$$N + R + C3 = 10 * C2 + E$$

$$E + O + C2 = 10 * C1 + N$$

$$S + M + C1 = 10 * M + O$$
(3)

A nível de código, estas contas serão expressas por duas equações distintas, separando o valor correspondente a um algarismo da resposta final do valor do *carry* por meio do uso das expressões de divisão e módulo, como indicado nas equações abaixo.

$$E = (N + R + C3) \div 10$$

$$C2 = (N + R + C3) \mod 10$$
(4)

Vale notar que o mesmo é verdade para o problema POINT + ZERO = ENERGY, como descrito nos conjuntos de equações a seguir.

1

$$O, I, N, T, R, G, Y \in [0, 9]$$

 $E, Z, P \in [1, 9]$ (5

$$P \neq O \neq I \neq N \neq T \neq Z \neq E \neq R \neq G \neq Y \qquad (6)$$

$$T + O = 10 * C4 + Y$$

$$N + R + C4 = 10 * C3 + G$$

$$I + E + C3 = 10 * C2 + R$$

$$O + Z + C2 = 10 * C1 + E$$

$$P + C1 = 10 * E + N$$
(7)

Com isso definido, pode-se estruturar o software e adaptar tais regras para a linguagem Prolog.

III. ESTRUTURA DO SOFTWARE

A estruturação do código desenvolvido se deu pela transformação das equações acima em predicados, que são utilizados como regras para que o software chegue a uma resolução.

Primeiramente, o conjunto de equações 1 pode ser realizado por meio do predicado between/3 [2], que faz um vínculo da variável, determinando limites superior e inferior aos seus possíveis valores. Assim, como exemplo, utiliza-se o comando between(0, 9, D) para se definir que D é um algarismo de 0 a 9.

Então, a equação 2 pode ser expressa por um conjunto de expressões similares. No caso, em Prolog, cria-se um predicado para todas as inequalidades de cada uma das variáveis. Dessa forma, tendo duas variáveis definidas, pode-se marcálas como diferentes ao se utilizar o operador \setminus =. É necessário criar uma regra distinta para cada dupla de variáveis, como, por exemplo, YM, YS, YŌ, YR, YN, YD, YĒ.

Finalmente, as equações de determinação dos valores do resultado da soma, expressas em 4, são determinadas de maneira similar às formas já escritas. Com isso, para o caso utilizado como exemplo na equação em questão, define-se as restrições E is (N+R+C3) mod 10 e C2 is (N+R+C3) div 10.

Quanto à ordem dos predicados, nota-se que esta propriedade do código é de suma importância para a eficiência do tempo de execução. Visando minimizar este tempo, seguiu-se a estratégia de se definir os valores na mesma ordem em que eles seriam resolvidos por uma pessoa.

Entre os valores, nota-se que é essencial se definir primeiro as restrições de existência das variáveis (neste caso, a relação definida pelo predicado between/3), para então se definir outras restrições e, por fim, as relações matemáticas entre os valores.

Dessa forma, segue-se a seguinte ordem:

- Define-se as restrições de existência das variáveis D e E com o predicado predicado between/3;
- Define-se a inequalidade entre as variáveis D e E;
- Define-se os valores de C3 (igual a $(D+E) \div 10$) e Y (igual a $(D+E) \mod 10$).

A partir de então, é descrito o próximo conjunto de variáveis, agora levando-se em consideração a presença de *carries* e a necessidade de se garantir a inequalidade de cada nova variável com todas as variáveis anteriores.

O código final para a resolução deste problema pode ser visto no bloco de código abaixo.

```
C1 C2 C3
     S E N D
     MORE+
 * M O N E Y
smm_puzzle(S, E, N, D, M, O, R, Y) :-
  /* Começando as constraints pelos algarismos

→ menos significativos */
  /* D e E são algarismos diferentes entre si */
 between (0, 9, D),
 between(0, 9, E),
 D = E
  /* Fazendo a soma e passando o carry para a

→ variável C3 */

  /*D + E = 10*C3 + Y */
 C3 is (D+E) div 10,
 Y is (D+E) \mod 10,
  /* N e R também são algarismos diferentes
  \hookrightarrow entre si,
   * e diferentes de D e E */
 between(0, 9, N),
 N=D, N=E,
 between (0, 9, R),
 R=N, R=D, R=E,
  /* Fazendo a soma e passando o carry para a
  → variável C2 */
  /* N + R + C3 = 10*C2 + E */
 C2 is (N+R+C3) div 10,
 E is (N+R+C3) \mod 10,
  /* O é algarismo diferente dos já
  → inicializados */
 between(0, 9, 0),
 O\=R, O\=N, O\=D, O\=E,
  /* Fazendo a soma e passando o carry para a

→ variável C1 */

  /* E + O + C2 = 10*C1 + N */
 C1 is (E+O+C2) div 10,
 N is (E+O+C2) \mod 10,
  /* S e M são algarismos diferentes dos já
  \hookrightarrow inicializados.
   * Ambos devem ser diferentes de 0,
   * porque números não começam com 0. */
 between (1, 9, S),
 S=0, S=R, S=N, S=D, S=E,
 between(1, 9, M),
 M = S, M = O, M = R, M = N, M = D, M = E,
  /* Y também deve ser diferente dos outros
  → algarismos */
 between(0, 9, Y),
 Y = M, Y = S, Y = O, Y = R, Y = N, Y = D, Y = E,
  /* Fazendo a soma e passando o carry para a

→ variável M */

  /* S + M + C1 = 10*M + 0 */
 M is (S+M+C1) div 10,
 O is (S+M+C1) mod 10.
```

O mesmo processo é realizado para o problema POINT + ZERO = ENERGY. Neste, vale notar que algumas condições diferentes precisam ser especificadas; por exemplo, como E é o primeiro algarismo de ENERGY, este deve ser inicializado com valor mínimo igual a 1, mesmo que sua inicialização seja

realizada antes dessa informação ser relevante. Isso garante uma maior eficiência à execução do programa.

O código final para a resolução deste problema pode ser visto no bloco de código abaixo.

```
C1 C2 C3 C4
     P O I N T
        Z E R O +
 * E N E R G Y
pze_puzzle(P, O, I, N, T, Z, E, R, G, Y) :-
 /* Começando as constraints pelos algarismos
  → menos significativos */
  /* T e O são algarismos diferentes entre si */
 between(0, 9, T),
 between(0, 9, 0),
 T = 0.
  /* Fazendo a soma e passando o carry para a

→ variável C4 */

  /* T + O = 10*C4 + Y
 C4 is (T+O) div 10,
 Y is (T+O) mod 10,
  /* N e R também são algarismos diferentes

→ entre si.

   * e diferentes de T e O */
 between(0, 9, N),
 N=T, N=0,
 between(0, 9, R),
 R=N, R=T, R=0,
  /* Fazendo a soma e passando o carry para a
  → variável C3 */
  /* N + R + C4 = 10*C3 + G */
 C3 is (N+R+C4) div 10,
 G is (N+R+C4) \mod 10,
  /* I e E são algarismos diferentes dos já
  \hookrightarrow inicializados
  * E também deve ser diferente de 0,
   * porque números não começam com 0. */
 between(0, 9, I),
 I = R, I = N, I = T, I = O,
 between (1, 9, E),
 E = I, E = R, E = N, E = T, E = O,
  /* Fazendo a soma e passando o carry para a

→ variável C2 */

  /* I + E + C3 = 10*C2 + R */
 C2 is (I+E+C3) div 10,
 R is (I+E+C3) \mod 10,
  /* Z é um algarismo diferente dos já
  * Também deve ser diferente de 0,
  * porque números não começam com 0.
  * O já foi inicializado. */
 between(1, 9, \mathbb{Z}),
 Z = E, Z = I, Z = R, Z = N, Z = I, Z = O,
  /* Fazendo a soma e passando o carry para a
     variável C1 */
  /* 0 + Z + C2 = 10*C1 + E */
 C1 is (O+Z+C2) div 10,
 E is (O+Z+C2) \mod 10,
  /* P é um algarismo diferente dos já

→ inicializados.

  * Também deve ser diferente de 0,
   * porque números não começam com 0. */
 between(1, 9, P),
 P = Z, P = E, P = I, P = R, P = N, P = T, P = O,
  /* G e Y também devem ser diferentes dos
  → outros algarismos */
 between (0, 9, G),
 G = P, G = Z, G = E, G = I, G = R, G = N, G = T,
  \hookrightarrow G\=0,
```

```
between(0, 9, Y),

Y\=G, Y\=P, Y\=Z, Y\=E, Y\=I, Y\=R, Y\=N,

→ Y\=T, Y\=O,

/* Fazendo a soma e passando o carry para a

→ variável E */

/* P + C1 = 10*E + N */

E is (P+C1) div 10,

N is (P+C1) mod 10.
```

IV. DESCRIÇÃO DOS EXPERIMENTOS

Como experimento para conferir a eficácia dos códigos desenvolvidos, foi criado também um código com uma organização diferente, de tal maneira que a execução do código fosse lenta. Especificamente, para ambos os códigos, foi utilizada uma organização tal que todos os predicados between/3 fossem definidos primeiro; depois, todas as inequalidades relevantes; e, por fim, as somas essenciais.

Como exemplo, pode-se ver o código realizado nesse experimento para o puzzle POINT + ZERO = ENERGY, com os comentários redundantes removidos, no bloco abaixo.

```
pze_puzzle_ocd(P, O, I, N, T, Z, E, R, G, Y) :-
  /* Betweens */
 between (0, 9, T),
 between(0, 9, 0),
 between(0, 9, N),
 between (0, 9, R),
 between(0, 9, I),
 between (1, 9, E),
 between (1, 9, Z),
 between(1, 9, P),
 between (0, 9, G),
 between (0, 9, Y),
  /* Inequalidades */
 T = 0,
 N=T, N=O,
 R=N, R=T, R=0,
 I = R, I = N, I = T, I = O,
 E = I, E = R, E = N, E = T, E = O,
 Z = E, Z = I, Z = R, Z = N, Z = I, Z = O,
 P = Z, P = E, P = I, P = R, P = N, P = T, P = O,
 G = P, G = Z, G = E, G = I, G = R, G = N, G = T,

    G\=0.

 Y = G, Y = P, Y = Z, Y = E, Y = I, Y = R, Y = N,
  \hookrightarrow Y\=T, Y\=O,
  /* Somas */
 C4 is (T+O) div 10,
 Y is (T+O) mod 10,
 C3 is (N+R+C4) div 10,
 G is (N+R+C4) \mod 10,
 C2 is (I+E+C3) div 10,
 R is (I+E+C3) \mod 10,
 C1 is (O+Z+C2) div 10,
 E is (O+Z+C2) \mod 10,
 E is (P+C1) div 10,
 N is (P+C1) mod 10.
```

V. ANÁLISE DOS RESULTADOS

A execução dos programas principais, organizados de modo a melhorar a eficiência do problema, foram executados com sucesso. No caso do problema SEND + MORE = MONEY, encontrou-se a única solução correta – D=7, E=5, M=1, N=6, O=0, R=8, S=9, eY=2 – em apenas 0.023

segundos, como pode-se ver na Figura 3. A solução é ilustrada na Figura 4.

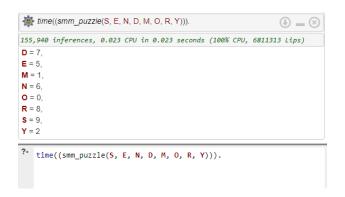


Fig. 3. Resolução do problema SEND + MORE = MONEY.

Fig. 4. Resultado final do problema SEND + MORE = MONEY.

No caso do problema POINT + ZERO = ENERGY, também encontrou-se a única solução correta – E=1, G=7, I=5, N=0, O=8, P=9, R=6, T=4, Y=2, eZ=3 – em 0.221 segundos, como pode-se ver na Figura 5.

```
time((pze puzzle(P, O, I, N, T, Z, E, R, G, Y))).
1,499,510 inferences, 0.212 CPU in 0.212 seconds (100% CPU, 70
60287 Lips)
E = 1
G = 7
I = 5,
N = 0
0 = 8
P = 9.
R = 6
T = 4.
Y = 2.
Z = 3
                                              0.213 seconds cpu time
Next
       10 | 100 | 1,000
                         Stop
?- time((pze_puzzle(P, O, I, N, T, Z, E, R, G, Y))).
```

Fig. 5. Resolução do problema POINT + ZERO = ENERGY.

Já os casos de teste mal organizados, como esperado tiveram um péssimo resultado em termos de performance. O código do puzzle SEND + MORE = MONEY foi concluído após 49.599 segundos de execução (Figura 6), enquanto que o código do problema POINT + ZERO = ENERGY excedeu o tempo limite do copilador utilizado (Figura 7).

```
## time((smm_puzzle_ocd(S, E, N, D, M, O, R, Y))).

$\rightarrow$ = \text{$\sigma$} \text{ time((smm_puzzle_ocd(S, E, N, D, M, O, R, Y)))}.

$\rightarrow$ = \text{$\sigma$} \
```

Fig. 6. Teste não eficiente do problema SEND + MORE = MONEY.

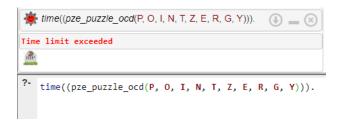


Fig. 7. Teste não eficiente do problema POINT + ZERO = ENERGY.

VI. CONCLUSÕES E TRABALHOS FUTUROS

Com isso, conclui-se a execução da atividade proposta. Foram desenvolvidos dois códigos distintos para a resolução de dois problemas cripto-aritméticos distintos com o uso da linguagem Prolog.

O exercício foi muito útil, em particular devido à experiência adquirida com esta linguagem de programação tão importante no cenário atual de Inteligência Artificial.

Como trabalhos futuros em relação aos códigos desenvolvidos, pode-se citar um maior foco em performance, alterando mais a organização dos predicados no código de maneira a maximizar a eficiência do processamento, ou até alterar o conjunto de predicados referentes à inequalidade entre todas as variáveis de cada problema de cripto-aritmética, de maneira que se melhore o rendimento e a entendibilidade do programa desenvolvido.

REFERENCES

- [1] COLMERAUER, Alain. ROUSSEL, Philippe. *The birth of Prolog*. Disponível em: http://alain.colmerauer.free.fr/alcol/ArchivesPublications/PrologHistory/19november92.pdf. Acesso em: 18 out. 2022.
- [2] SWI-Prolog between/3. Disponível em: https://www.swi-prolog.org/pl doc/man?predicate=between/3. Acesso em: 18 out. 2022.