Introdução à Lógica

a partir de sua história filosófica

Volume 2

De Aristóteles a C++

Alvaro Pereira Pires
Ana Luiza de Almeida e Sousa
Anne Caroline Froes Amaral
Paulo Virgílio Lemes Aguiar
Raquel Anna Sapunaru

Há uma grande diferença entre usar uma língua e sistematizar a sua gramática. Há também uma diferença, igualmente grande, entre usar raciocínios dedutivos e sistematizá-los. Até onde se sabe, foi Aristóteles que atentou para esse fato pela primeira vez. Porém, ele não se ateve a demarcar explicitamente a validade desses raciocínios, de modo necessariamente exato. Aristóteles deu conta de validades e invalidades de um tipo muito restrito de raciocínios e a este, no texto "Analíticos Anteriores", Livro I, doravante AALI, dedicou sua busca pela verdade. Trata-se dos silogismos. Nosso trabalho consiste em implementar através de rotinas escritas em C++, alguns silogismos aristotélicos apresentados em AAL1, fazendo também uso da classificação medieval conforme apresentado no livro *Deductive Logic* de St. G. W. J. Stock. Nem todos os capítulos ou exemplos de AAL1 foram contemplados, principalmente porque a certa altura Aristóteles se repete.

9 786559 170722

editora φfi.org

Introdução à Lógica a partir de sua história filosófica

Introdução à Lógica
a partir de sua história filosófica

Volume 2

De Aristóteles a C++

Alvaro Pereira Pires
Ana Luiza de Almeida e Sousa
Anne Caroline Froes Amaral
Paulo Virgílio Lemes Aguiar
Raquel Anna Sapunaru

Diagramação: Marcelo A. S. Alves
Capa: Lucas Margoni

O padrão ortográfico e o sistema de citações e referências bibliográficas são prerrogativas de cada autor. Da mesma forma, o conteúdo de cada capítulo é de inteira e exclusiva responsabilidade de seu respectivo autor.

Dados Internacionais de Catalogação na Publicação (CIP)

PIRES, Alvaro Pereira; et al (Orgs.)

Introdução à Lógica a partir de sua história filosófica, volume 2: de Aristóteles a C++ [recurso eletrônico] / Alvaro Pereira Pires; et al (Orgs.) -- Porto Alegre, RS: Editora Fi, 2021.

86 p.

ISBN - 978-65-5917-072-2
DOI - 10.22350/9786559170722

Disponível em: http://www.editorafi.org

1. Filosofia; 2. História; 3. Lógica; 4. Clássica; 5. Introdução; I. Título.

CDD: 100

Índices para catálogo sistemático:
1. Filosofia 100

Agradecimentos
ao CNPQ pela Bolsa PIBIC

Sumário

Seção 1

Introdução

Há uma grande diferença entre usar uma língua e sistematizar a sua gramática. Há também uma diferença, igualmente grande, entre usar raciocínios dedutivos e sistematizá-los. Até onde se sabe, foi Aristóteles que atentou para esse fato pela primeira vez. Porém, ele não se ateve a demarcar explicitamente a validade desses raciocínios, de modo necessariamente exato. Aristóteles deu conta de validades e invalidades de um tipo muito restrito de raciocínios e a este, no texto "Analíticos Anteriores", Livro I, doravante *AALI*, dedicou sua busca pela verdade. Trata-se dos silogismos.

Aristóteles era filho de um médico, natural de Estagira. Aristóteles ingressou na Academia de Platão, perto do seu décimo sétimo aniversário, em 367 a.C., e por quase vinte anos foi seguidor dos pensamentos de Platão. Após a morte de Platão, em 347 a.C., partiu de Atenas e, ficou distante de lá por quase doze anos. Um dos motivos de sua partida foi a questão política que se instalou na época. Entretanto, após as leis macedônias tomarem Atenas, Aristóteles retornou e lá permaneceu por mais doze anos, dando continuidade aos seus estudos. Em 342 a.C., tornou-se tutor do Alexandre "o Grande", mas em 323 a.C., seu discípulo veio a falecer. Pouco tempo depois, o filósofo foi acusado por irreverência religiosa e, mais uma vez, deixou Atenas. Em Calsis, veio a falecer na propriedade da sua mãe.

Sua escola, denominada Liceu, conhecida também como escola peripatética, foi comparada à Academia de Platão, e provavelmente só foi fundada após sua morte. Entretanto, Aristóteles criou um programa de pesquisa, uma biblioteca de mapas e também um museu de história natural, em conjunto com seus discípulos. Devido a esses fatos, a filosofia

aristotélica foi instituída como base para a matemática, astronomia, medicina e também para a história da filosofia natural grega.

Retornando à lógica aristotélica, esta que mais influenciou o pensamento lógico europeu, da Idade Média até a segunda metade do século XIX, mesmo que a lógica estoica demonstrasse resultados mais relevantes. Uma falha pasmosa da lógica de Aristóteles, citada por inúmeros lógicos e historiadores da lógica, é lidar com a frase "Todo o grego é europeu" como se esta derivasse do acréscimo de um quantificador, "Todo", a uma sentença que tem uma estrutura lógica idêntica a de "Sócrates é europeu", o que é visivelmente falso. Em "Sócrates é europeu", "é europeu" é o predicado de "Sócrates", o sujeito da sentença. Porém, em "Todo o grego é europeu", "grego" não é verdadeiramente o sujeito da sentença. O sujeito, que está oculto só se aparece na lógica contemporânea, que especifica a forma lógica de "Todo o grego é europeu" como "Todo o sujeito que for grego é também europeu". Dizer "Todo o grego é europeu" não é o mesmo que designar o predicado "europeu" ao sujeito "grego", mas designar o predicado "europeu" a todo o sujeito que tiver o predicado "grego". Então, para conceber a lógica aos moldes de Aristóteles "grego" passa a ser o *termo sujeito*, que são na verdade predicados que ocupam o lugar sintático dos sujeitos verdadeiros. Nos *AALI*, Aristóteles repete constantemente que o *termo sujeito* pode trocar de lugar, ou melhor dizendo se converter[1] ao predicado, mas isso não ocorre com os sujeitos das sentenças usualmente formuladas. Dizer que "Sócrates é grego" não é o mesmo que dizer que "Grego é Sócrates" e não se trata de estilo de expressão. Há casos nos quais é possível a conversão do sujeito em predicado, mas isso não ocorre sempre.

Nosso trabalho consiste em implementar através de rotinas escritas em C++, alguns silogismos aristotélicos apresentados em *AAL1*, fazendo também uso da classificação medieval conforme apresentado no livro

1 (a) Conversão direta: troca-se o sujeito pelo predicado e vice-versa. Por exemplo: Todo *mortal* é *homem* Todo *homem* é *mortal*; e; (b) Conversão acidental: a premissa tem seu sujeito e predicado trocados entre si. Por exemplo: *Todo homem* é *mortal* *Algum mortal* é *homem.*

Deductive Logic de St. G. W. J. Stock. Nem todos os capítulos ou exemplos de *AAL1* foram contemplados, principalmente porque a certa altura Aristóteles se repete.

Seção 2

AAL1 - I

Partiremos de algumas definições, a saber:

a) "Constuma-se usar a palavra "proposição" para designar o significado de uma sentença [...]" (COPI, 1981, p. 22).
b) "[...] Um argumento é qualquer grupo de proposições tal que se afirme ser uma delas derivada das outras, as quais são consideradas provas evidentes da verdade da primeira." (COPI, 1981, p. 23).
c) "A conclusão de um argumento é aquela proposição que se afirma com a base nas outras proposições desse mesmo argumento, e, por sua vez, essas outras proposições que são enunciados como prova ou razões para aceitar a conclusão são as premissas desse argumento." (COPI, 1981, p. 23).

Para Aristóteles, a proposição universal se aplica a tudo ou a nada; a proposição particular se aplica a alguma coisa e a proposição indefinida não se refere a universalidade, tampouco a particularidade. Assim sendo, sejam:

proposições universais:

Todo C é A
Nenhum B é A

proposições particulares:

Algum C é A
Algum D não é A

proposições indefinidas:

C é A
B não é A

Para Aristóteles, "O silogismo é uma locução em que, uma vez certas suposições sejam feitas, alguma coisa distinta delas se segue necessariamente devido à mera presença das suposições como tais." (AAL1, 24b20); e o silogismo perfeito é "o que nada requer além do que

nele está compreendido para evidenciar a necessária conclusão [...]" (AAL1, 24b25).

Todo C é A
Nenhum B é A

AAL1 – I - Portugol

“Contrários são objeto da mesma ciência.”

“B e C é A”

Código:

```
Início
   inteiro A, B, C, conjuntoA[10],conjuntoB[4], conjuntoC[5], BigualA=0,
CigualA=0
   para A=0; A<10; A++
      conjuntoA[A] = A
para B=0; B<4; B++
      conjuntoB[B] = 6 + B
para C=0; C<5; C++
      conjuntoC[C] = C
para B=0; B<4; B++
      para A=0; A<10; A++
         caso conjuntoB[B]==conjuntoA[A]
                  BigualA++
para C=0; C<5; C++
      para A=0; A<5; A++
         caso conjuntoC[C] = conjuntoA[A]
            CigualA++
caso BigualA =4 e CigualA =5
      escrever "B e C é A"
Fim
```

“O prazer não é bem.”

“B não é C”

Código:

```
Início
   inteiro B, C, conjuntoB[4], conjuntoC[5], igual=0
   para B=0; B<4; B++
      conjuntoB[B] = 6 + B
para C=0; C<5; C++
      conjuntoC[C] = C;
para B=0; B<4; B++
      para C=0; C<5; C++
         caso conjuntoB[B] = conjuntoC[C]
            igual++;
caso igual = 0
      escrever "B não é C"
Fim
```

AAL1 – I - Código C++

“Contrários são objeto da mesma ciência.”

“B e C é A”

A={0,1,2,3,4,5,6,7,8,9}

C={0,1,2,3,4}

B={6,7,8,9}

D={0,1,6,9}

Código:

```
#include <stdio.h>
#include <locale.h>
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
   int A, B, C, conjuntoA[10],conjuntoB[4], conjuntoC[5], BigualA=0,
CigualA=0;
   for(A=0; A<10; A++){
      conjuntoA[A] = A;
   }
   for(B=0; B<4; B++){
      conjuntoB[B] = 6 + B;
   }
     for(C=0; C<5; C++){
      conjuntoC[C] = C;
   }
   for(B=0; B<4; B++){
      for(A=0; A<10; A++){
         if (conjuntoB[B]==conjuntoA[A])
         BigualA++;
```

```
      }
   }
   for(C=0; C<5; C++){
      for(A=0; A<5; A++){
         if (conjuntoC[C]==conjuntoA[A])
         CigualA++;
      }
   }
  if(BigualA==4 && CigualA==5)
      printf("B e C é A");
   return 0;
}
```

“O prazer não é bem.”

“B não é C”

Código:

```
#include <stdio.h>
#include <locale.h>
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
   int B, C, conjuntoB[4],conjuntoC[5], igual=0;
   for(B=0; B<4; B++){
      conjuntoB[B] = 6 + B;
   }
   for(C=0; C<5; C++){
      conjuntoC[C] = C;
   }
   for(B=0; B<4; B++){
      for(C=0; C<5; C++){
         if (conjuntoB[B]==conjuntoC[C])
         igual++;
```

```
        }
    }
   if(igual == 0)
        printf("B não é C");
    return 0;
}
```

AAL1 – II

“Se nenhum prazer é bem, tampouco será uma coisa boa, prazer.”

Nessa premissa considera-se um universal negativo e convertível nos seus próprios termos. Em “Ser universal” é feita uma generalização para um todo, sem especificações, negativa porque nega e convertível pelo fato dos seus próprios termos deixarem claro o que é preciso ser expressado. Porém, em:

“Se todo prazer é bem, algum bem tem também que ser prazer.”

A premissa é considerada universal afirmativa, mas a sua conversão é feita em uma particular. Na primeira parte da proposição a caracterização de universal é feita com a palavra “todo”, uma generalização de “prazer”, mas já na sua conversão a particularidade é especificada com a palavra “algum”. Portanto, a partir da universal convertemos a proposição em uma particular, partindo de um prazer como um todo e distinguindo algum tipo de prazer.

“Se todo B é animal e A, homem, posto que homem não se aplica a todo animal, porém todo animal se aplicar a todo homem).”

Então, “animal” e “homem” são definidos como termos nessa proposição, de forma que se consiga mostrar que se A não se aplica a B, não necessariamente B não se aplica em A. Logo, fica evidenciado que o homem é visto em sua totalidade como um animal, mas o animal não é visto em sua totalidade como um homem, pois o animal engloba vários tipos de seres, não só o homem.

AAL1 – II - Portugol

“Se nenhum prazer é bem, tampouco será uma coisa boa, prazer.”
“Nenhum B é C, Nenhum B é D, Todo C é D”

C = 0,1,2,3,4
B = 6,7,8,9
D= 0,1,2,3,4,5
Código:

```
Início
   inteiro B, C, D, conjuntoB[4],conjuntoC[5],conjuntoD[6],
AlgumBeC=0, AlgumBeD=0, TodoCeD=0
para B=0; B<4; B++
      conjuntoB[B] = 6 + B
para C=0; C<5; C++
      conjuntoC[C] = C
para D=0; D<6; D++
      conjuntoD[D] = D
para B=0; B<4; B++
      para C=0; C<5; C++
         caso conjuntoB[B] = conjuntoC[C])
            AlgumBeC++
para B=0; B<4; B++
      para D=0; D<6; D++
         caso conjuntoB[B]==conjuntoD[D]
            AlgumBeD++
para C=0; C<5; C++
      para (D=0; D<6; D++
         caso conjuntoC[C] = conjuntoD[D]
            TodoCeD++
```

```
caso AlgumBeC = 0 e AlgumBeD = 0 e TodoCeD e 5
      escrever "Nenhum B é C, Nenhum B é D, Todo C é D"
Fim
```

“Se todo prazer é bem, algum bem tem também que ser prazer.”
“Todo B é A, Algum A é B”

Código

```
Início
   inteiro A, B, conjuntoA[10],conjuntoB[4], AlgumAeB=0, BigualA=0
   para A=0; A<10; A++
      conjuntoA[A] = A
para B=0; B<4; B
      conjuntoB[B] = 6 + B
para B=0; B<4; B++
      para A=0; A<10; A++
         caso conjuntoB[B] = conjuntoA[A]
            BigualA++
para A=0; A<10; A++
      para (B=0; B<4; B++
         caso conjuntoA[A] = conjuntoB[B]
            AlgumAeB++
caso BigualA = 4 e AlgumAeB >0
      escrever "Todo B é A, Algum A é B"
Fim
```

“Se todo B é animal e A, homem, posto que homem não se aplica a todo animal, porém animal se aplica a todo homem.”
“Algum A é B, todo B é A”

Código:

```
Início
   inteiro A, B, conjuntoA[10],conjuntoB[4], AlgumAeB=0, BigualA=0
   para A=0; A<10; A++
```

```
        conjuntoA[A] = A
para B=0; B<4; B++
        conjuntoB[B] = 6 + B
para A=0; A<10; A++
        para B=0; B<4; B++
            caso conjuntoA[A] =conjuntoB[B]
                AlgumAeB++
para B=0; B<4; B++
        para A=0; A<10; A++
            caso conjuntoB[B] = conjuntoA[A]
                BigualA++
caso AlgumAeB>0 e BigualA=4
        escrever "Algum  A é B, todo B é A"
Fim
```

AAL1 – II - Código C++

“Se nenhum prazer é bem, tampouco será uma coisa boa, prazer.”
“Nenhum B é C
Nenhum B é D
Todo C é D”
C={0,1,2,3,4}
B={6,7,8,9}
D={0,1,2,3,4,5}
Código:

```
#include <stdio.h>
#include <locale.h>
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
   int B, C, D, conjuntoB[4],conjuntoC[5],conjuntoD[6], AlgumBeC=0,
AlgumBeD=0, TodoCeD=0;
   for(B=0; B<4; B++){
      conjuntoB[B] = 6 + B;
   }
   for(C=0; C<5; C++){
      conjuntoC[C] = C;
   }
   for(D=0; D<6; D++){
      conjuntoD[D] = D;
   }
   for(B=0; B<4; B++){
      for(C=0; C<5; C++){
         if (conjuntoB[B]==conjuntoC[C])
```

```
            AlgumBeC++;
        }
    }
    for(B=0; B<4; B++){
        for(D=0; D<6; D++){
            if (conjuntoB[B]==conjuntoD[D])
            AlgumBeD++;
        }
    }
    for(C=0; C<5; C++){
        for(D=0; D<6; D++){
            if (conjuntoC[C]==conjuntoD[D])
            TodoCeD++;
        }
    }
 if(AlgumBeC==0 && AlgumBeD==0 && TodoCeD==5)
        printf("Nenhum B é C, Nenhum B é D, Todo C é D");
    return 0;
}
```

“Se todo prazer é bem, algum bem tem também que ser prazer.”
“Todo B é A, Algum A é B”

Código

```
    int A, B, conjuntoA[10],conjuntoB[4], AlgumAeB=0, BigualA=0;
    for(A=0; A<10; A++){
        conjuntoA[A] = A;
    }
    for(B=0; B<4; B++){
        conjuntoB[B] = 6 + B;
    }
    for(B=0; B<4; B++){
        for(A=0; A<10; A++){
```

```
        if (conjuntoB[B]==conjuntoA[A])
        BigualA++;
      }
   }
   for(A=0; A<10; A++){
      for(B=0; B<4; B++){
        if (conjuntoA[A]==conjuntoB[B])
        AlgumAeB++;
      }
   }
 if(BigualA==4 && AlgumAeB>0)
      printf("Todo B é A, Algum A é B");
   return 0;
}
```

“Se todo B é animal e A, homem, posto que homem não se aplica a todo animal, porém animal se aplica a todo homem.”

“Algum A é B, todo B é A”

Código:

```
#include <stdio.h>
#include <locale.h>
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
        int A, B, conjuntoA[10],conjuntoB[4], AlgumAeB=0, BigualA=0;
   for(A=0; A<10; A++){
      conjuntoA[A] = A;
   }
   for(B=0; B<4; B++){
      conjuntoB[B] = 6 + B;
   }
   for(A=0; A<10; A++){
```

```
        for(B=0; B<4; B++){
            if (conjuntoA[A]==conjuntoB[B])
            AlgumAeB++;
        }
    }
    for(B=0; B<4; B++){
        for(A=0; A<10; A++){
            if (conjuntoB[B]==conjuntoA[A])
            BigualA++;
        }
    }
if( AlgumAeB>0 && BigualA==4)
        printf("Algum  A é B, todo B é A");
    return 0;
}
```

AAL1 - III

“Que um homem não é um cavalo.”

Esta proposição é dita contingente devido ao fato que, o predicado, um cavalo, necessariamente não se aplica ao sujeito, o homem. A questão da contingência nas proposições está relacionada com o caso dela ser necessariamente verdadeira ou não ser necessariamente verdadeira. Desta forma, as premissas negativas se convertem por si só. Se A não se aplica a nenhum B, B não se aplica a nenhum A. Com isso, é contingente dizer que homem não se aplica a nenhum cavalo. Do mesmo modo, é contingente que cavalo não se aplique a nenhum homem.

“Branco não se aplica a nenhuma vestimenta.”

O raciocínio desta premissa segue a mesma lógica da anterior. Entretanto, neste caso, o predicado não se aplica ao sujeito. Por exemplo, se a vestimenta é branca, não necessariamente quer dizer que todas as vestimentas são brancas. Entretanto, existem casos de a vestimenta ser branca. A isso Aristóteles chama-se de contingência.

“É não branco.”

Quando diz-se “é contingente que A não se aplique a nenhum B” percebe-se que “é contingente” expressa uma ideia de afirmação, por ser uma consequência da palavra “é”, que em qualquer situação faz afirmação. Por isso, “É não branco”, fornece uma afirmação da proposição.

AAL1 – III – Portugol

“Que um homem não é um cavalo”

```
Início
        Função Principal
        inteiro n
        escrever “Que um homem não é um cavalo”
        escrever “Insira um número inteiro positivo”
        leia n
        se n%2=0
                escrever “o número par, será sempre par”
        se não
                escrever “o número ímpar, será sempre ímpar”
Fim
```

“Branco não se aplica a nenhuma vestimenta”

```
Início
   Função Principal
   inteiro n,d = 2,Primo=1
   escrever "Branco não se aplica a nenhuma vestimenta";
   escrever "Todo número primo, exceto o 2, é ímpar. Mas nem todo
número ímpar é primo"
   escrever "Insira um número inteiro positivo diferente de 2"
   leia "%d"
   se n%2 != 0
      escrever "é ímpar"
    se não
```

```
        escrever "é par"

    para d=2; d<n; d++
       se  n%d == 0
          pause;
    se == n
         escrever " e primo.");
    se não
         escrever "e não é primo."
Fim
```

“É não branco.”

```
Início
Função Principal
inteiro n;
       escrever "Que um homem não e um cavalo";
       escrever "Insira um número inteiro positivo"
       leia "d"
  se n % 2 == 0
       escrever "é não ímpar"
  se não
       escrever "\n é não par \n "
Fim
```

AAL1 – III - Código C++

“Que um homem não é um cavalo.”

“Um número par será sempre par”

“Um número ímpar será sempre ímpar”

Código:

```
 #include <stdio.h>
 #include <locale.h>
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
         int n;
      printf("Que um homem não é um cavalo. \n ");
      printf("Insira um número inteiro positivo: \n ");
      scanf("%d", &n);
   if (n % 2 == 0){
      printf ("\n um número par, será sempre par \n ");
   }else{
      printf("\n um número ímpar, será sempre ímpar \n ");
       }
   return 0;
       }
```

“Branco não se aplica a nenhuma vestimenta”

“Todo número primo, exceto o 2, é ímpar. Mas nem todo número ímpar é primo.”

Código:

```
 #include <stdio.h>
#include <locale.h>
```

```
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
         int n,d = 2,Primo=1;
   printf("Insira um número inteiro positivo diferente de 2: \n ");
   scanf("%d", &n);
   if (n%2 != 0) {
       printf("é ímpar");
   } else {
       printf("é par");
         }
   for (d=2; d<n; d++)
      if (n%d == 0)
         break;
   if (d==n) {
       printf(" e primo.");
   } else {
       printf(" e não é primo.");
         }
}
```

“É não branco.”

Insere um número e testa:

“É não ímpar”

“É não par”

Código:

```
#include <stdio.h>
#include <locale.h>
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
      int n;
```

```
    printf("Insira um número inteiro positivo: \n ");
    scanf("%d", &n);
  if (n % 2 == 0){
    printf ("\n é não ímpar \n ");
  }else{
    printf("\n é não par \n ");
     }
  return 0;
}
```

AAL1 – IV

“Nenhum homem é pedra
Algum animal é homem
Algum animal não é pedra.”

Esse silogismo permite inferir diversas interpretações sobre o que foi dito. Sabe-se que nenhum homem é pedra e que algum animal é homem. Com isso, pode-se chegar a conclusão silogística de que algum animal que é homem não pode ser pedra, ou seja, sabe-se que pedra não se aplica ao homem.

“Toda medicina é ciência
Toda linha é medicina
Toda linha é ciência.”

Nesse silogismo o raciocínio lógico é mais rápido e fácil de se entender. Sabendo que a medicina é um tipo de ciência e que toda linha é medicina, chegamos à conclusão silogística de que toda linha é um tipo de ciência. A conclusão neste silogismo é mais perceptível que no anterior, sendo a interpretação mais clara.

AAL1 – IV – Portugol

“Nenhum homem é pedra
Algum animal é homem
Algum animal não é pedra.”

Nenhum A é B
Algum C é A
Algum C não é B
A = {0,1,2,3,4}
B = {5,6,7,8,9}
C = {3,4,5}
Início
Função Principal

```
   inteiro H, P, A,conjuntoH[5], conjuntoP[5], conjuntoA[3], j, igual=0;
  //definindo o conjunto que irá representar Homem, conjuntoM
   para H = 0; H < 5; H++

      conjuntoH[H]=H++;
  //definindo o conjunto que irá representar Pedra, conjuntoP
   para  P = 5;P < 10; P++
         conjuntoP[P]=P++
   //definindo o conjunto que irá representar Animal, conjuntoA
   para A = 3; A < 6; A++
      conjuntoA[A]=A
   para H=0; H<5; H++
      para P=5; P<10; P++
         se conjuntoH[H] =! conjuntoP[P]
            igual++
   se igual > 0
```

```
      escrever "Nenhum homem é pedra"
   se não
   escrever "Erro"
    para H=0; H<5; H++
       para A=3; A<6; A++
          se conjuntoH[H] == conjuntoA[A]
             igual++
   se igual > 0
     escrever "Algum animal é homem"
   se não
     escrever "Erro\n"
   escrever "Algum animal não é pedra”
Fim
```

“Toda medicina é ciência
Toda linha é medicina
Toda linha é ciência.”

```
Todo B é A
Todo C é B
Todo C é A
A = {0,1,2,3,4,5}
B = {2,3,4,5}
C = {2,3,4}
Função Principal
   inteiro c, m, L,conjuntoc[10], conjuntom[5], conjuntoL[3], j, igual=0
  //definindo o conjunto que irá representar Ciência, conjuntoc
   para c = 0; c < 10; c++

      conjuntoc[c]=c++;
  //definindo o conjunto que irá representar Medicina, conjuntom
   para m = 0;m < 5; m++
      conjuntom[m]=2*m;
```

```
    //definindo o conjunto que irá representar Linha, conjuntoL
    para (L = 0; L < 3; L++)
       conjuntoL[L]=L*4;
    para c=0; c<5; c++
       para m=0; m<5; m++
          se conjuntoc[c] == conjuntom[m]
             igual++
    se igual > 0
      escrever "Toda medicina é ciência"
    se não
    escrever "Erro"
     para m=0; m<5; m++
       para L=0; L<3; L++
          se conjuntom[m] == conjuntoL[L]
             igual++
    se igual > 0
      escrever "Toda linha é medicina"
    se não
      escrever "Erro"
    escrever "Toda linha é ciência"
Fim
```

AAL1 – IV - Código C++

“Nenhum homem é pedra
Algum animal é homem
Algum animal não é pedra.”

Nenhum A é B
Algum C é A
Algum C não é B
A = {0,1,2,3,4}
B = {5,6,7,8,9}
C = {3,4,5}
Código:

```
 #include <stdio.h>
#include <locale.h>
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
      int a, b, c,conjuntoA[5], conjuntoB[5], conjuntoC[3], igual=0;
   for (a = 0; a < 5; a++)
   {
      conjuntoA[a]=a;
   }
   for (b = 0;b < 5; b++)
   {
      conjuntoB[b]=5 + b;
   }
    for (c = 0; c < 3; c++)
   {
      conjuntoC[c]= 3 + c;
```

```
    }
    for (a=0; a<5; a++) {
        for (b=0; b<5; b++) {
            if (conjuntoA[a] == conjuntoB[b])
                igual++;
        }
    }
    if (igual == 0)
      printf("Nenhum A é B\n");
    else {
    printf("Erro\n");
    }
    igual=0;
     for (a=0; a<5; a++) {
        for (c=0; c<3; c++) {
            if (conjuntoC[c] == conjuntoA[a])
                igual++;
        }
    }
    if (igual > 0)
      printf("Algum C é A\n");
    else {
      printf("Erro\n");
       }
    printf("Algum C não é B\n");
    return 0;
}
```

“Toda medicina é ciência
Toda linha é medicina
Toda linha é ciência.”
Todo B é A

Todo C é B

Todo C é A

A = {0,1,2,3,4,5}

B = {2,3,4,5}

C = {2,3,4}

Código:

```
#include <stdio.h>
#include <locale.h>
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
   int a, b, c,conjuntoA[6], conjuntoB[4], conjuntoC[3], igual=0;
   for (a = 0; a < 6; a++)
   {
      conjuntoA[a]=a;
   }
    for (b = 0;b < 4; b++)
   {
      conjuntoB[b]= 2 + b;
   }
   for (c = 0; c < 3; c++)
   {
      conjuntoC[c]=2 + c;
   }
   for (a=0; a<6; a++) {
      for (b=0; b<4; b++) {
         if (conjuntoB[b] == conjuntoA[a])
            igual++;
      }
   }
   if (igual == 4)
     printf("Todo B é A \n");
```

```
    else {
    printf("Erro\n");
    }
    igual=0;
     for (b=0; b<4; b++) {
        for (c=0; c<3; c++) {
            if (conjuntoC[c] == conjuntoB[b])
                igual++;
        }
    }
    if (igual == 3)
      printf("Todo C é B\n");
    else {
      printf("Erro\n");
    }
    printf("Todo C é A \n");
    return 0;
}
```

AAL1 – IX

“Todo homem é animal
João é homem
João é animal.”

Neste primeiro silogismo do livro IX, é visto duas maneiras de interpretação sem perder a lógica na primeira e terceira premissa, ou seja, podem ser escritas na forma de afirmação ou negação com a presença da palavra necessariamente. Desse modo, todo homem é homem animal, João é homem, portanto João é animal. Já a outra maneira, João não é necessariamente um animal, João é homem logo, João não é necessariamente um animal. O silogismo assume a seguinte forma:

B é(não) nA
C é B
C é(não) nA

“Todo animal é movente
Todo homem é animal
Todo homem é movente.”

Este silogismo apresenta apenas um formato, diferentemente do anterior. Sabe-se que todo animal é movente, todo homem é animal sendo assim, todo homem é movente. O silogismo assume a seguinte forma:

Todo B é A
Todo C é B
Todo C é A

“Não é necessário que todo animal é movente
Não é necessário que todo homem é animal
Não é necessário que todo homem é movente.”

Observa-se que este silogismo faz a negação do exemplo anterior. Por consequência, não é necessário que todo animal é movente e que todo homem é animal com isso, conclui-se que não é necessário que todo homem é movente. O silogismo assume a seguinte forma:

Não é necessário que todo B é A
Não é necessário que todo C é B
Não é necessário que todo C é A

“Todo cachorro é necessariamente canino
Algum Bob é cachorro
Algum Bob é necessariamente canino.”

Neste silogismo é visto mais uma vez a presença da palavra novamente na primeira e terceira premissa. Entretanto, o mesmo assume apenas um formato. Com isso, todo cachorro é necessariamente canino, algum Bob é cachorro e, consequentemente, algum Bob é necessariamente canino. O silogismo assume a seguinte forma:

Todo B é necessariamente A
Algum C é B
Algum C é necessariamente A

“Não é necessário que todo animal é movente
Algum branco é animal
Algum branco é movente.”

No último exemplo do livro IX, repete-se o termo “necessário” compondo o silogismo. Dessa forma, o mesmo assume o formato de que não é necessário que todo animal é movente, algum branco é animal então, algum branco é movente. É válido ressaltar que todos os exemplos deste livro pertencem ao Grupo 1 da Gramática da Lógica. O silogismo assume a seguinte forma:

Não é necessário que todo B é A
Algum C é B
Não é necessário que algum C é A

AAL1 – IX – Portugol

“Todo homem é animal
João é homem
João é animal.”

B é(não) A
C é B
C é(não) A
A{0,1,2,3,4}
B{0,1,2,3,4}
C{0,1,2,3,4}
Início
Portugol

```
setlocale LC_ALL, "Portuguese_Brazil"
inteiro a, b, c,conjuntoA[5], conjuntoB[5], conjuntoC[5], j, igual=0
para a = 0; a < 5; a++
    conjuntoA[a]=a;
    escrever "%i ",conjuntoA[a]
    escrever "\n";
para b = 0;b < 5; b++
    conjuntoB[b]=b
    escrever "%i ",conjuntoB[b])
   escrever "\n"
para c = 0; c < 5; c++
    conjuntoC[c]=c;
    escrever "%i ",conjuntoC[c])
    escrever "\n")
enquanto igual == 0 && j < 5
    se conjuntoA[j] == conjuntoB[j]
```

```
      igual = 0;
      j++
      se não
      igual = 1;
      se igual == 0
        escrever "\nB é A"
      se não
        escrever "\nErro\n"
   enquanto igual == 0 && j < 5
     se  conjuntoC[j] == conjuntoB[j]

      igual = 0;
      j++
      se não
      igual = 1
      se igual == 0
        escrever "\nC é B"
      se não
        escrever "\nErro\n"
  enquanto igual == 0 && j < 5
     se conjuntoC[j] == conjuntoA[j]
      igual = 0
      j++
      se não
      igual = 1
      se igual == 0
        escrever "\nC é A")
      se não
        escrever "\nErro\n")
Fim
```

“Todo animal é movente

Todo homem é animal

Todo homem é movente.”

```
Todo B é A
Todo C é B
Todo C é A
A = {0,1,2,3,4}
B = {2,3,4}
C = {5,6,7,8}
Início
Função Principal
   Setlocale LC_ALL, "Portuguese_Brazil"
   inteiro a, b, c,conjuntoA[5], conjuntoB[4], conjuntoC[3], igual=0
   para a = 0; a < 5; a++
      conjuntoA[a]=a;
      escrever "%i ",conjuntoA[a])
      escrever "\n"
   para b = 0;b < 4; b++
      conjuntoB[b]=b
      escrever "%i ",conjuntoB[b]
      escrever "\n"
   para c = 0; c < 3; c++
      conjuntoC[c]=c;
      escrever "%i ",conjuntoC[c])
      escrever "\n"
   para a=0; a<5; a++
      para b=0; b<4; b++
         se conjuntoB[b] == conjuntoA[a])
            igual++
      se igual == 4)
        escrever "\nTodo B é A"
      se não
```

```
        escrever "\nErro\n"

  igual=0
   para b=0; b<4; b++
      para c=0; c<3; c++
         se conjuntoC[c] == conjuntoB[b]
            igual++
      se igual == 3
         escrever "\nTodo C é B"
      se não
         escrever "\nErro\n"
 igual=0
    para a=0; a<5; a++
      para c=0; c<3; c++
         se conjuntoC[c] == conjuntoA[a]
            igual++
      se igual == 3
         escrever "\nTodo C é A\n")
      se não
         escrever "\nErro\n"
Fim
```

“Não é necessário que todo animal é movente
Não é necessário que todo homem é animal
Não é necessário que todo homem é movente.”

Não é necessário que todo B é A
Não é necessário que todo C é B
Não é necessário que todo C é A
A = {0,1,2,3,4}
B = {2,3,4,5,6}
C = {4,5,6,7,8}
Início

```
Função Principal
   setlocale LC_ALL, "Portuguese_Brazil"
   inteiro a, b, c,conjuntoA[5], conjuntoB[5], conjuntoC[5], igual=0
   para a = 0; a < 5; a++
      conjuntoA[a]=a;
      escrever "%i ",conjuntoA[a]
      escrever "\n")
   para b = 0;b < 5; b++
      conjuntoB[b]=2+b
      escrever "%i ",conjuntoB[b])
      escrever "\n"
   para c = 0; c < 5; c++
      conjuntoC[c]=4+c
      escrever "%i ",conjuntoC[c])
      escrever "\n")
   para a=0; a<5; a++
      para b=0; b<5; b++
         se conjuntoB[b] == conjuntoA[a]
            igual++
      se igual >= 0
        escrever "\n Não é necessário que todo B é A"
      se não
        escrever "\nErro\n"
  igual=0
  para c=0; c<5; c++
      para b=0; b<5; b++
         se conjuntoB[b] == conjuntoC[c]
            igual++
      se igual >= 0
         escrever "\n Não é necessário que todo C é B"
      se não
         escrever "\nErro\n")
```

```
  igual=0
    para a=0; a<5; a++
      para c=0; c<5; c++
        se conjuntoC[c] == conjuntoA[a]
          igual++
      se igual >= 0
        escrever "\n Não é necessário que todo C é A"
      se não
        escrever "\nErro\n"
Fim
```

“Todo cachorro é necessariamente canino
Algum Bob é cachorro
Algum Bob é necessariamente canino.”

Todo B é necessariamente A
Algum C é B
Algum C é necessariamente A
A = {0,1,2,3,4}
B = {2,3,4}
C = {4,5,6,7,8}

```
Início
Função Principal
   setlocale LC_ALL, "Portuguese_Brazil"
   inteiro a, b, c,conjuntoA[5], conjuntoB[3], conjuntoC[5], igual=0
   para a = 0; a < 5; a++
      conjuntoA[a]=a
      escrever "%i ",conjuntoA[a])
      escrever "\n")
   para b = 0;b < 3; b++
      conjuntoB[b]=2+b
      escrever "%i ",conjuntoB[b]
```

```
        escrever "\n"
    para c = 0; c < 5; c++
        conjuntoC[c]=4+c
        escrever "%i ",conjuntoC[c])
        escrever "\n"
    para a=0; a<5; a++
        para b=0; b<3; b++
            se conjuntoB[b] == conjuntoA[a]
                igual++
        se igual == 3
            escrever "\n Todo B é necessariamente A"
        se não
            escrever "\nErro\n"
    igual=0
    para c=0; c<5; c++
        para b=0; b<3; b++
            se conjuntoB[b] == conjuntoC[c]
                igual++
        se igual > 0
            escrever "\n Algum C é B"
        se não
            escrever "\nErro\n"
    igual=0
      para a=0; a<5; a++
        for c=0; c<3; c++
            se conjuntoC[c] == conjuntoA[a]
                igual++
        se igual > 0
            escrever "\n Algum C é necessariamente A"
        se não
            escrever "\nErro\n"
Fim
```

“Não é necessário que todo animal é movente
Algum branco é animal
Algum branco é movente.”

```
Não é necessário que todo B é A
Algum C é B
Não é necessário que algum C é A
A = {0,1,2,3,4}
B = {2,3,4,5,6}
C = {5,6,7,8,9}
Início
Função Principal
   setlocale LC_ALL, "Portuguese_Brazil"
   inteiro a, b, c,conjuntoA[5], conjuntoB[5], conjuntoC[5], igual=0
   para a = 0; a < 5; a++
      conjuntoA[a]=a
      escrever "%i ",conjuntoA[a]
      escrever "\n"
   para b = 0;b < 5; b++
      conjuntoB[b]=2+b
      escrever "%i ",conjuntoB[b])
      escrever "\n"
   para c = 0; c < 5; c++
      conjuntoC[c]=5+c
      escrever "%i ",conjuntoC[c]
      escrever "\n"
   para a=0; a<5; a++
      para b=0; b<5; b++
         se conjuntoB[b] == conjuntoA[a]
            igual++
      se igual >= 0
        escrever "\nNão é necessário que todo B é A"
```

```
      se não
        escrever "\nErro\n"
  igual=0
   para c=0; c<5; c++
      para b=0; b<5; b++
         se conjuntoB[b] == conjuntoC[c]
            igual++
      se igual > 0
         escrever "\n Algum C é B"
      se não
         escrever "\nErro\n"

  igual=0
    para a=0; a<5; a++
      para c=0; c<5; c++
         se conjuntoC[c] == conjuntoA[a]
            igual++
      se igual >= 0
         escrever "\n Não é necessário que algum C é A"
      se não
         escrever "\nErro\n"
Fim
```

AAL1 – IX - Código C++

“Todo homem é animal
João é homem
João é animal.”

B é(não) A
C é B
C é(não) A
A{0,1,2,3,4}
B{0,1,2,3,4}
C{0,1,2,3,4}
Código:

```
#include <stdio.h>
#include <locale.h>
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
   int a, b, c,conjuntoA[5], conjuntoB[5], conjuntoC[5], j, igual=0;
   for (a = 0; a < 5; a++)
   {
      conjuntoA[a]=a;
      printf("%i ",conjuntoA[a]);
   }
      printf("\n");
   for (b = 0;b < 5; b++)
   {
      conjuntoB[b]=b;
      printf("%i ",conjuntoB[b]);
   }
```

```
   printf("\n");
for (c = 0; c < 5; c++)
{
   conjuntoC[c]=c;
   printf("%i ",conjuntoC[c]);
}
   printf("\n");
while(igual == 0 && j < 5)
{
   if (conjuntoA[j] == conjuntoB[j])
{
   igual = 0;
   j++;
}
   else
   igual = 1;
}
   if (igual == 0)
      printf("\nB é A");
   else {
      printf("\nErro\n");
        }
while(igual == 0 && j < 5)
{
   if (conjuntoC[j] == conjuntoB[j])
{
   igual = 0;
   j++;
}
   else
   igual = 1;
}
```

```
        if (igual == 0)
           printf("\nC é B");
        else {
           printf("\nErro\n");
            }
  while(igual == 0 && j < 5)
   {
      if (conjuntoC[j] == conjuntoA[j])
   {
      igual = 0;
      j++;
   }
      else
      igual = 1;
   }
      if (igual == 0)
         printf("\nC é A");
      else {
         printf("\nErro\n");
          }
   return 0;
}
```

“Todo animal é movente
Todo homem é animal
Todo homem é movente.”

Todo B é A

Todo C é B

Todo C é A

A = {0,1,2,3,4}

B = {2,3,4}

C = {5,6,7,8}

Código:

```
#include <stdio.h>
#include <locale.h>
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
   int a, b, c,conjuntoA[5], conjuntoB[4], conjuntoC[3], igual=0;
   for (a = 0; a < 5; a++)
   {
      conjuntoA[a]=a;
      printf("%i ",conjuntoA[a]);
   }
      printf("\n");
   for (b = 0;b < 4; b++)
   {
      conjuntoB[b]=b;
      printf("%i ",conjuntoB[b]);
   }
      printf("\n");
   for (c = 0; c < 3; c++)
   {
      conjuntoC[c]=c;
      printf("%i ",conjuntoC[c]);
   }
      printf("\n");
   for (a=0; a<5; a++) {
      for (b=0; b<4; b++) {
         if (conjuntoB[b] == conjuntoA[a])
            igual++;
      }
   }
      if (igual == 4)
```

```
        printf("\nTodo B é A");
      else {
        printf("\nErro\n");
         }
  igual=0;
   for (b=0; b<4; b++) {
      for (c=0; c<3; c++) {
        if (conjuntoC[c] == conjuntoB[b])
           igual++;
      }
   }
      if (igual == 3)
        printf("\nTodo C é B");
      else {
        printf("\nErro\n");
         }
 igual=0;
    for (a=0; a<5; a++) {
      for (c=0; c<3; c++) {
        if (conjuntoC[c] == conjuntoA[a])
           igual++;
      }
   }
      if (igual == 3)
        printf("\nTodo C é A\n");
      else {
        printf("\nErro\n");
         }
   return 0;
}
```

“Não é necessário que todo animal é movente
Não é necessário que todo homem é animal
Não é necessário que todo homem é movente.”

```
Não é necessário que todo B é A
Não é necessário que todo C é B
Não é necessário que todo C é A
A = {0,1,2,3,4}
B = {2,3,4,5,6}
C = {4,5,6,7,8}
Início
Função Principal;
   setlocale LC_ALL, "Portuguese_Brazil"
   inteiro a, b, c,conjuntoA[5], conjuntoB[5], conjuntoC[5], igual=0
   para a = 0; a < 5; a++
      conjuntoA[a]=a;
      escrever "%i ",conjuntoA[a])
      escrever "\n"
   para b = 0;b < 5; b++
      conjuntoB[b]=2+b
      escrever "%i ",conjuntoB[b])
      escrever "\n"
   para c = 0; c < 5; c++
      conjuntoC[c]=4+c
      escrever "%i ",conjuntoC[c])
      escrever "\n"
   para a=0; a<5; a++
      para b=0; b<5; b++
         se conjuntoB[b] == conjuntoA[a])
            igual++
      se igual >= 0
        escrever "\n Não é necessário que todo B é A"
      se não
```

```
        escrever "\nErro\n"
  igual=0
  para c=0; c<5; c++
      para b=0; b<5; b++
          se conjuntoB[b] == conjuntoC[c])
              igual++
      se igual >= 0
          escrever "\n Não é necessário que todo C é B"
      se não
          escrever "\nErro\n")
 igual=0
    para a=0; a<5; a++
      para c=0; c<5; c++
          se conjuntoC[c] == conjuntoA[a])
              igual++
      se igual >= 0
          escrever "\n Não é necessário que todo C é A"
      se não
          escrever "\nErro\n"
Fim
```

“Todo cachorro é necessariamente canino
Algum Bob é cachorro
Algum Bob é necessariamente canino.”

Todo B é necessariamente A
Algum C é B
Algum C é necessariamente A
A = {0,1,2,3,4}
B = {2,3,4}
C = {4,5,6,7,8}
Código:

```
#include <stdio.h>
```

```
#include <locale.h>
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
   int a, b, c,conjuntoA[5], conjuntoB[3], conjuntoC[5], igual=0;
   for (a = 0; a < 5; a++)
   {
      conjuntoA[a]=a;
      printf("%i ",conjuntoA[a]);
   }
      printf("\n");
   for (b = 0;b < 3; b++)
   {
      conjuntoB[b]=2+b;
      printf("%i ",conjuntoB[b]);
   }
      printf("\n");
   for (c = 0; c < 5; c++)
   {
      conjuntoC[c]=4+c;
      printf("%i ",conjuntoC[c]);
   }
      printf("\n");
   for (a=0; a<5; a++) {
      for (b=0; b<3; b++) {
         if (conjuntoB[b] == conjuntoA[a])
            igual++;
      }
   }
      if (igual == 3)
        printf("\n Todo B é necessariamente A");
      else {
```

```
        printf("\nErro\n");
         }
  igual=0;
   for (c=0; c<5; c++) {
      for (b=0; b<3; b++) {
         if (conjuntoB[b] == conjuntoC[c])
            igual++;
      }
   }
      if (igual > 0 )
         printf("\n Algum C é B");
      else {
         printf("\nErro\n");
          }
 igual=0;
    for (a=0; a<5; a++) {
      for (c=0; c<3; c++) {
         if (conjuntoC[c] == conjuntoA[a])
            igual++;
      }
   }
      if (igual > 0)
         printf("\n Algum C é necessariamente A");
      else {
         printf("\nErro\n");
          }

   return 0;
}
```

“Não é necessário que todo animal é movente
Algum branco é animal
Algum branco é movente.”

Não é necessário que todo B é A
Algum C é B
Não é necessário que algum C é A
A = {0,1,2,3,4}
B = {2,3,4,5,6}
C = {5,6,7,8,9}
Código:

```
#include <stdio.h>
#include <locale.h>
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
   int a, b, c,conjuntoA[5], conjuntoB[5], conjuntoC[5], igual=0;
   for (a = 0; a < 5; a++)
   {
      conjuntoA[a]=a;
      printf("%i ",conjuntoA[a]);
   }
      printf("\n");
   for (b = 0;b < 5; b++)
   {
      conjuntoB[b]=2+b;
      printf("%i ",conjuntoB[b]);
   }
      printf("\n");
   for (c = 0; c < 5; c++)
   {
      conjuntoC[c]=5+c;
      printf("%i ",conjuntoC[c]);
```

```
  }
     printf("\n");
  for (a=0; a<5; a++) {
     for (b=0; b<5; b++) {
        if (conjuntoB[b] == conjuntoA[a])
           igual++;
     }
  }
     if (igual >= 0)
       printf("\n Não é necessário que todo B é A");
     else {
       printf("\nErro\n");
        }
 igual=0;
  for (c=0; c<5; c++) {
     for (b=0; b<5; b++) {
        if (conjuntoB[b] == conjuntoC[c])
           igual++;
     }
  }
     if (igual > 0 )
        printf("\n Algum C é B");
     else {
        printf("\nErro\n");
        }
igual=0;
   for (a=0; a<5; a++) {
     for (c=0; c<5; c++) {
        if (conjuntoC[c] == conjuntoA[a])
           igual++;
     }
  }
```

```
    if (igual >= 0)
       printf("\n Não é necessário que algum C é A");
    else {
       printf("\nErro\n");
        }
  return 0;
}
```

AAL1 – X

“Nenhum cachorro é felino
Todo gato é felino
Nenhum gato é cachorro.”

Nos exemplos do livro X, é visto que todos compõem o Grupo 2 da Gramática da Lógica dos nomes silogísticos. Este exemplo é um deles. A primeira premissa é universal negativa, segunda universal negativa e a última universal negativa. Sendo assim, nenhum cachorro é felino, todo gato é felino então, nenhum gato é cachorro. O silogismo assume a seguinte forma:
Nenhum B é A
Todo C é A
Nenhum C é B

“Nenhum cachorro é felino
Alguma Mimi é felino
Alguma Mimi não é cachorro.”

O segundo exemplo do Grupo 2, ao contrário do anterior, tem a presença de proposições particulares. A primeira premissa é universal negativa, a segunda particular positiva e a terceira universal positiva. Portanto, nenhum cachorro é felino, alguma Mimi é felino, com isso, alguma Mimi não é cachorro. O silogismo assume a seguinte forma:
Nenhum A é B
Algum C é B
Algum C é A

“Todo cachorro é canino
Nenhum Paulo é canino
Nenhum Paulo é cachorro.”

No último exemplo deste livro, a primeira premissa é universal positiva, a segunda universal negativa e último universal negativa. Por fim, todo cachorro é canino, nenhum Paulo é canino com isso, nenhum Paulo é cachorro. O silogismo assume a seguinte forma:
Todo B é A
Nenhum C é A
Nenhum C é B

AAL1 – X – Portugol
“Nenhum cachorro é felino
Todo gato é felino
Nenhum gato é cachorro.”

```
Nenhum B é A
Todo C é A
Nenhum C é B
A = {0,1,2,3,4}
B = {6,7,8,9}
C = {2,3,4}
Início
Função Principal
   setlocale LC_ALL, "Portuguese_Brazil"
   inteiro a, b, c,conjuntoA[5], conjuntoB[4], conjuntoC[3], igual=0
   para a = 0; a < 5; a++
      conjuntoA[a]=a
      escrever "%i ",conjuntoA[a]
      escrever "\n"
   para b = 0;b < 4; b++
      conjuntoB[b]=6+b
```

```
    printf "%i ",conjuntoB[b]
    printf "\n"
  para c = 0; c < 3; c++
    conjuntoC[c]=2+c
    printf "%i ",conjuntoC[c]
    printf "\n"
  para a=0; a<5; a++
    para b=0; b<3; b++
      se conjuntoB[b] == conjuntoA[a]
        igual++
    se igual == 0
      escrever "\n Nenum B é A"
    se não
      printf "\nErro\n"
 igual=0
 para a=0; a<5; a++
    para c=0; c<3; c++
      se conjuntoC[c] == conjuntoA[a]
        igual++
    se igual == 3
      escrever "\nTodo C é A"
    se não
      escrever "\nErro\n"

igual=0
  para b=0; b<4; b++
    para c=0; c<3; c++
      se conjuntoC[c] == conjuntoB[b]
        igual++
    se igual == 0
      escrever "\n Nenhum C é B\n"
    se não
```

```
        escrever "\nErro\n"
Fim
```

“Nenhum cachorro é felino
Alguma Mimi é felino
Alguma Mimi não é cachorro.”

```
Nenhum A é B
Algum C é B
Algum C é A
A = {0,1,2,3,4}
B = {5,6,7,8,9}
C = {3,4,5,6,7}
Início
Função Principal
   setlocale LC_ALL, "Portuguese_Brazil"
   inteiro a, b, c,conjuntoA[5], conjuntoB[5], conjuntoC[5], igual=0
   para a = 0; a < 5; a++
      conjuntoA[a]=a
      printf("%i ",conjuntoA[a])
      escrever "\n"
   para b = 0;b < 5; b++
      conjuntoB[b]=5+b
      escrever "%i ",conjuntoB[b]
         escrever "\n"
   para c = 0; c < 5; c++
      conjuntoC[c]=3+c
      printf "%i ",conjuntoC[c]
      printf "\n"
   para a=0; a<5; a++
      para b=0; b<5; b++
         se conjuntoB[b] == conjuntoA[a]
            igual++
```

```
      se igual == 0
        escrever "\nNenhum A é B"
      se não
        escrever "\nErro\n"

  igual=0
  para b=0; b<5; b++
      para c=0; c<5; c++
        se conjuntoC[c] == conjuntoB[b]
          igual++
      se igual > 0
        escrever "\nAlgum C é B"
      se não
        printf "\nErro\n"

 igual=0
   para a=0; a<5; a++
      para c=0; c<3; c++
        se conjuntoC[c] == conjuntoA[a]
          igual++
      se igual > 0
        escrever "\nAlgum C é A\n")
      se não
        escrever "\nErro\n"
Fim
```

“Todo cachorro é canino
Nenhum Paulo é canino
Nenhum Paulo é cachorro.”
Todo B é A
Nenhum C é A
Nenhum C é B

```
A = {0,1,2,3,4}
B = {2,3,4}
C = {5,6,7,8,9}
Início
Função Principal
   setlocale LC_ALL, "Portuguese_Brazil"
   inteiro a, b, c,conjuntoA[5], conjuntoB[3], conjuntoC[5], igual=0
   para a = 0; a < 5; a++
      conjuntoA[a]=a
      escrever "%i ",conjuntoA[a]
      escrever "\n"
   para b = 0;b < 3; b++
      conjuntoB[b]=2+b
      escrever "%i ",conjuntoB[b]
      escrever "\n"
   para c = 0; c < 5; c++
      conjuntoC[c]=5+c
      escrever "%i ",conjuntoC[c]
      escrever "\n"
   para a=0; a<5; a++
      para b=0; b<3; b++
         se conjuntoB[b] == conjuntoA[a]
            igual++
      se igual == 3
        escrever "\n Todo B é A"
      se não
        escrever "\nErro\n"
  igual=0
  para a=0; a<5; a++
      para c=0; c<5; c++
         se conjuntoC[c] == conjuntoA[a]
            igual++
```

```
        se igual == 0
            escrever "\nNenhum C é A"
        se não
            escrever "\nErro\n"
  igual=0;
     para c=0; c<5; c++
        para b=0; b<3; b++
            se conjuntoB[b] == conjuntoC[c]
                igual++
        se igual == 0
            escrever "\n Nenhum C é B\n"
        se não
            escrever"\nErro\n"
Fim
```

AAL1 – X - Código C++

“Nenhum cachorro é felino
Todo gato é felino
Nenhum gato é cachorro.”

Nenhum B é A
Todo C é A
Nenhum C é B
A = {0,1,2,3,4}
B = {6,7,8,9}
C = {2,3,4}
Código:

```
#include <stdio.h>
#include <locale.h>
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
   int a, b, c,conjuntoA[5], conjuntoB[4], conjuntoC[3], igual=0;
   for (a = 0; a < 5; a++)
   {
      conjuntoA[a]=a;
      printf("%i ",conjuntoA[a]);
   }
      printf("\n");
   for (b = 0;b < 4; b++)
   {
      conjuntoB[b]=6+b;
      printf("%i ",conjuntoB[b]);
   }
```

```
    printf("\n");
  for (c = 0; c < 3; c++)
  {
    conjuntoC[c]=2+c;
    printf("%i ",conjuntoC[c]);
  }
    printf("\n");
  for (a=0; a<5; a++) {
    for (b=0; b<3; b++) {
      if (conjuntoB[b] == conjuntoA[a])
        igual++;
    }
  }
    if (igual == 0)
      printf("\n Nenum B é A");
    else {
      printf("\nErro\n");
       }
 igual=0;
  for (a=0; a<5; a++) {
    for (c=0; c<3; c++) {
      if (conjuntoC[c] == conjuntoA[a])
        igual++;
    }
  }
    if (igual == 3 )
      printf("\nTodo C é A");
    else {
      printf("\nErro\n");
       }
 igual=0;
   for (b=0; b<4; b++) {
```

```
      for (c=0; c<3; c++) {
         if (conjuntoC[c] == conjuntoB[b])
            igual++;
      }
   }
      if (igual == 0)
         printf("\n Nenhum C é B\n");
      else {
         printf("\nErro\n");
          }
   return 0;
}
```

“Nenhum cachorro é felino
Alguma Mimi é felino
Alguma Mimi não é cachorro.”

Nenhum A é B
Algum C é B
Algum C é A
A = {0,1,2,3,4}
B = {5,6,7,8,9}
C = {3,4,5,6,7}
Código:

```
#include <stdio.h>
#include <locale.h>
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
   int a, b, c,conjuntoA[5], conjuntoB[5], conjuntoC[5], igual=0;
   for (a = 0; a < 5; a++)
   {
      conjuntoA[a]=a;
```

```
    printf("%i ",conjuntoA[a]);
  }
    printf("\n");
  for (b = 0;b < 5; b++)
  {
    conjuntoB[b]=5+b;
    printf("%i ",conjuntoB[b]);
  }
    printf("\n");
  for (c = 0; c < 5; c++)
  {
    conjuntoC[c]=3+c;
    printf("%i ",conjuntoC[c]);
  }
    printf("\n");
  for (a=0; a<5; a++) {
    for (b=0; b<5; b++) {
      if (conjuntoB[b] == conjuntoA[a])
        igual++;
    }
  }
    if (igual == 0)
      printf("\nNenhum A é B");
    else {
      printf("\nErro\n");
       }
 igual=0;
  for (b=0; b<5; b++) {
    for (c=0; c<5; c++) {
      if (conjuntoC[c] == conjuntoB[b])
        igual++;
    }
```

```
    }
        if (igual > 0)
            printf("\nAlgum C é B");
        else {
            printf("\nErro\n");
             }
  igual=0;
     for (a=0; a<5; a++) {
        for (c=0; c<3; c++) {
            if (conjuntoC[c] == conjuntoA[a])
                igual++;
        }
    }
        if (igual > 0)
            printf("\nAlgum C é A\n");
        else {
            printf("\nErro\n");
             }
    return 0;
}
```

“Todo cachorro é canino
Nenhum Paulo é canino
Nenhum Paulo é cachorro.”

Todo B é A
Nenhum C é A
Nenhum C é B
A = {0,1,2,3,4}
B = {2,3,4}
C = {5,6,7,8,9}
Código:

```
#include <stdio.h>
```

```
#include <locale.h>
int main()
{
   setlocale(LC_ALL, "Portuguese_Brazil");
   int a, b, c,conjuntoA[5], conjuntoB[3], conjuntoC[5], igual=0;
   for (a = 0; a < 5; a++)
   {
      conjuntoA[a]=a;
      printf("%i ",conjuntoA[a]);
   }
      printf("\n");
   for (b = 0;b < 3; b++)
   {
      conjuntoB[b]=2+b;
      printf("%i ",conjuntoB[b]);
   }
      printf("\n");
   for (c = 0; c < 5; c++)
   {
      conjuntoC[c]=5+c;
      printf("%i ",conjuntoC[c]);
   }
      printf("\n");
   for (a=0; a<5; a++) {
      for (b=0; b<3; b++) {
         if (conjuntoB[b] == conjuntoA[a])
            igual++;
      }
   }
      if (igual == 3)
        printf("\n Todo B é A");
      else {
```

```
        printf("\nErro\n");
         }
  igual=0;
   for (a=0; a<5; a++) {
      for (c=0; c<5; c++) {
         if (conjuntoC[c] == conjuntoA[a])
            igual++;
      }
   }
      if (igual == 0)
         printf("\nNenhum C é A");
      else {
         printf("\nErro\n");
          }
 igual=0;
    for (c=0; c<5; c++) {
      for (b=0; b<3; b++) {
         if (conjuntoB[b] == conjuntoC[c])
            igual++;
      }
   }
      if (igual == 0)
         printf("\n Nenhum C é B\n");
      else {
         printf("\nErro\n");
          }
   return 0;
}
```

Bibliografia

ABBAGNANO, N. **Dicionário de Filosofia**. Tradução Alfredo Bossi. São Paulo: Martins Fontes, 2003.

ARISTÓTELES. **Organon. Analíticos Anteriores**. Livro I. Tradução Edson Bini. São Paulo: EDIPRO, 2005.

BRANQUINHO, J.; MURCHO, D.; GOMES, N.G. **Enciclopédia de Termos Lógico-Filosóficos**. 2005.Disponível em: <https://philarchive.org/archive/JOOEDT>. Acesso: 15 de jan. de 2021.

COPI, I.M. **Introdução à lógica**. Tradução Álvaro Cabral. São Paulo: Editora Mestre Jou, 1981.

DICIONÁRIO DE BIOGRAFIAS CIENTÍFICAS. Volume 1. Organizador Charles Coulston Gillispie. Tradução Carlos Almeida Pereira, et al. Rio de Janeiro: Contraponto, 2007.

LAKATOS, E.M.; MARCONI, M.A. **Fundamentos de Metodologia Científica**. 6.ed. São Paulo: Atlas 2005.

McKEON, R. **Introduction to Aristotle**. Nova York: The Modern Library, 1947.

MORAIS,B.R.;et al. **Introdução à lógica a partir de sua história filosófica** . Volume 1. 2019. Disponível em: <https:/www.editorafi.org/718logica>. Acesso: 15 de jan. de 2021.

PEREIRA, S. L. **Linguagem C++**. São Paulo: FATEC, 1999.

ROSENTAL, M.; IUDIN, P. **Diccionario Filosofico**. Buenos Aires: Ediciones Universo, 1973.

RUSSELL, B. **História do Pensamento Ocidental**. 5ª. Edição. Rio de Janeiro: EDIOURO, 2001.

A Editora Fi é especializada na editoração, publicação e divulgação de pesquisa acadêmica/científica das humanidades, sob acesso aberto, produzida em parceria das mais diversas instituições de ensino superior no Brasil. Conheça nosso catálogo e siga as páginas oficiais nas principais redes sociais para acompanhar novos lançamentos e eventos.

www.editorafi.org
contato@editorafi.org